Technology
Arts Sciences
TH Köln

Die Ermittlung von Sehenswürdigkeiten mittels Linked Open Data am Beispiel der Stadt Köln

Bachelorarbeit

ausgearbeitet von

Sebastian Leuer

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang

Medieninformatik

Erster Prüfer: Prof. Dipl.-Des. Christian Noss
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Kristian Fischer
Technische Hochschule Köln

Gummersbach, im September 2016

Adressen: Sebastian Leuer
[redacted]

Prof. Dipl.-Des. Christian Noss
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
christian.noss@th-koeln.de

Prof. Dr. Kristian Fischer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
kristian.fischer@th-koeln.de

Kurzfassung

Das Suchen ist eine der, wenn nicht die am häufigsten durchgeführte Tätigkeit im Internet. Täglich werden Suchmaschinen mit Problemen aus aller Welt und aller Domänen befragt, in der Hoffnung, dass das Internet eine Lösung bereitstellt. Neben dem *Web der Dokumente*, welches überwiegend die großen Suchmaschinenhersteller wie Google und Microsoft durchsuchen, existiert auch das weniger bekannte *Web der Daten*. In diesem Teil des Internets werden Daten, keine Dokumente, in einem festen Format kodiert. Dadurch soll die Möglichkeit geschaffen werden, dass nicht nur Menschen, sondern auch Maschinen, diese Daten verarbeiten können. Die Daten enthalten untereinander Verlinkungen, weswegen man auch von *Linked Data* spricht. Mit der vom W3C standardisierten Abfragesprache SPARQL ist es möglich, diese Daten nach selbst definierten Kriterien abzufragen.

Diese Arbeit befasst sich mit der Entwicklung einer SPARQL Abfrage zur Ermittlung von Sehenswürdigkeiten in Köln. Anhand dieses Anwendungsbeispiels soll beschrieben werden, inwieweit *Linked Data* in der Lage ist, mit Problemen und Fragestellungen des Alltags umzugehen. Es wird sich zeigen, dass es grundsätzlich möglich ist, derartige Anwendungsszenarien mit Linked Data zu lösen. Ein umfassendes Suchergebnis, welches beispielsweise Reiseführer geben, konnte jedoch nicht erzielt werden. Grund dafür sind hauptsächlich, wie in dieser Arbeit dargelegt wird, inkonsistente Daten.
Um diese Beobachtung aufstellen zu können, wurden präzisere Suchkriterien für Sehenswürdigkeiten spezifiziert. Weitere Auffälligkeiten, die während der Entwicklung bemerkt wurden, wurden entsprechend dokumentiert.

Abstract

Searching the web is one of the most often, if not even the most often, performed actions in the Internet. Each day, search engines crawl the Internet to find solutions for problems from any part of the world and all kinds of domains. Besides the *web of documents*, which large search engines like Google and Microsoft utilize, there is also the less known *web of data*. This part of the Internet contains data that is structured in a fixed format whereas the web of documents primarily focusses on documents. By structuring the data, machines shall have a better chance to automatically process the data, just like humans are able to process the web of documents. In addition, data in the web of data may contain so-called links to other data elements. Hence, the data elements are also called *Linked Data*. The W3C established a standardized query language called SPARQL, which allows to search the data according to self-defined criteria.

The work at hand analyzes to what extent Linked Data is capable of solving day-to-day search queries and problems. Hence, this work first demonstrates the development of an exemplary SPARQL query to find touristically relevant sights in Cologne and, second, compares how the search results alters along the query development. Ultimately, the work shows that Linked Data is in principle capable of solving such use cases. However, the results cannot compete with the extent of a common travel guide due to inconsistencies in data, as the analysis reveals. As a prerequisite for the analysis, the work had to define precise criteria for what renders a touristically relevant sight. Lastly, whenever the development of the queries led to anomalies, the analysis will point them out appropriately.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

LOD Linked Open Data

OWL Web Ontology Language

PICS Platform for Internet Content Selection

RDF Resource Description Framework

RDFS Resource Description Framework Schema

URI Uniform Resource Identifier

W3C World Wide Web Consortium

1. Einleitung

1.1. Motivation

Vor 25 Jahren, am 6. August 1991, wurde die erste Webseite der Welt der Öffentlichkeit zugänglich gemacht. Auch heute noch ist diese Webseite unter `http://info.cern.ch/hypertext/WWW/TheProject.html` auffindbar [o.VoJ]. Ihr Erfinder war Tim Berners-Lee. Seitdem entwickelte sich das Internet zu einer heutzutage nicht mehr wegzudenkenden Technologie. Mit neuen Errungenschaften wurden für das Internet oder dessen Teile auch neue Namen erfunden. Zu den populärsten Beispielen gehören unter anderem: *Web 2.0*, *Internet der Dinge*, *Darknet* oder *Web der Daten*. Zu nahezu allen Lebenssituationen kann man heutzutage das Internet heranziehen. Sprüche wie „Das Internet vergisst nie“ oder „Das Internet weiß alles“ haben sich schon längst zu einer Lebensweisheit entwickelt. Doch wie viel weiß das Internet wirklich?

1.2. Zielsetzung

Anhand des anwendungsorientierten Beispiels, die Ermittlung von Sehenswürdigkeiten in Köln, soll sich in dieser Arbeit der oben gestellten Frage genähert werden. Wird beispielsweise die Suchmaschine Google nach „sehenswürdigkeiten köln“ befragt, werden bereits einige bekannte Punkte aufgelistet, wie Abbildung 1.1 zeigt. Doch neben den herkömmlichen Suchmaschinen gibt es noch eine weitere, der Allgemeinheit nicht allzu bekannte Ebene des Internets.
Diese Ebene ist der Ansatz für ein intelligenteres und besser informiertes Internet und nennt sich *Semantic Web* oder *Web of Data*. In dieser Arbeit steht das Semantic Web im Mittelpunkt. Realisiert wird es durch die Technologie namens Linked Data [BHBL09].

Ziel dieser Arbeit soll die Entwicklung einer Abfragelösung sein, mit welcher Sehenswürdigkeiten der Stadt Köln ermittelt werden können. Dabei treten unter anderem die folgenden Fragen auf, die im Verlauf der Arbeit beantwortet werden:

- Welche Art von Sehenswürdigkeiten sollen gefunden werden?

- Sind die erforderlichen Daten überhaupt vorhanden?
- In welchem Format liegen die Daten vor?
- Welchen Aufwand erfordert das Abfragen der Daten?

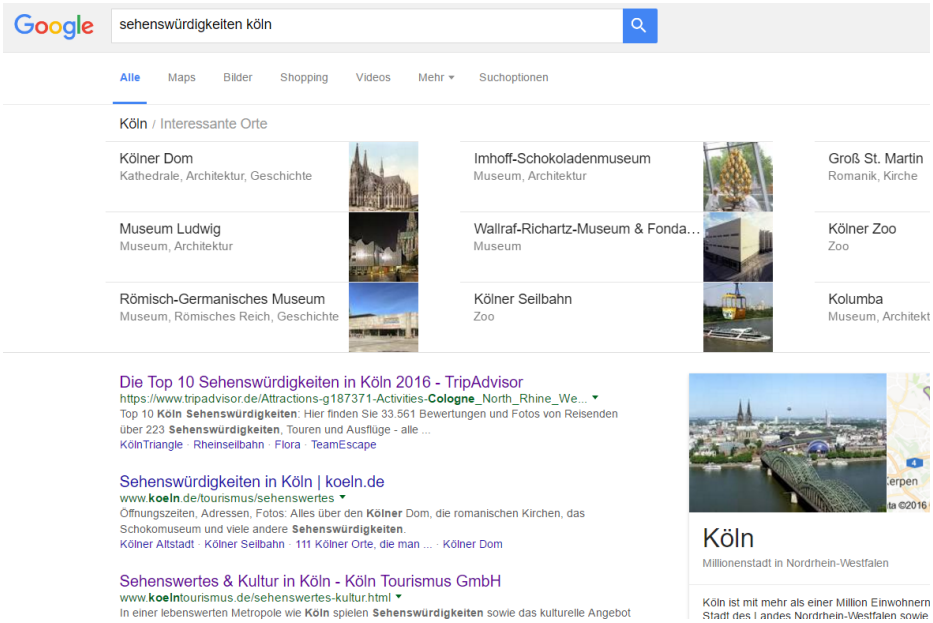

Abbildung 1.1.: Ausschnitt der Google-Suche nach Kölner Sehenswürdigkeiten (Stand: 22.08.2016)

1.3. Aufbau

Diese Bachelorarbeit ist im Folgendem in vier Abschnitte eingeteilt. In Kapitel 2 wird der Begriff der Sehenswürdigkeit definiert und bezogen auf die Stadt Köln genauer spezifiziert. Auf dieser Basis wird die Zielsetzung dieser Arbeit konkretisiert. In Kapitel 3 werden die technischen Grundlagen erläutert, die in dieser Arbeit zur Entwicklung der Abfrage verwendet werden. Eine genauere Beschreibung des Entwicklungsprozesses, der Abfrageresultate und welche Merkmale aufgefallen sind, erfolgt in Kapitel 4. Zum Abschluss wird in Kapitel 5 ein Ausblick auf künftige und unbeantwortete Fragen gegeben und ein Fazit gezogen.

2. Sehenswürdigkeiten in Köln

In diesem Kapitel soll der Begriff der Sehenswürdigkeit kurz etwas differenzierter beleuchtet werden. Es wird genauer betrachtet, was eine Sehenswürdigkeit ist und in welchem Kontext diese existieren. Es wird sich herausstellen, dass es viele verschiedene Interessengruppen, die jedoch nicht alle in dieser Arbeit berücksichtigt werden können. Daher werden im nächsten Schritt bestimmte Interessen vorgestellt und beispielhaft auf die Stadt Köln angewendet. Abschließend wird in diesem Kapitel die Zielsetzung dieser Arbeit konkretisiert.

2.1. Definition

Obwohl sich der Begriff „Sehenswürdigkeit“ selbst ganz gut beschreibt, macht es Sinn sich einmal ein paar Definitionen anzugucken. Dadurch können wichtige Eigenschaften besser herausgestellt werden.

So beschreibt der Duden eine Sehenswürdigkeit als „etwas wegen seiner Einmaligkeit, außergewöhnlichen Schönheit, Kuriosität o. Ä. besonders Sehenswertes, was nur an einem bestimmten Ort zu finden ist und deshalb besonders für Touristen von besonderem Interesse ist“ [Dud].

Eine ähnliche Definition liefert das Onlinewörterbuch thefreedictionary.com. Demnach ist eine Sehenswürdigkeit „ein Gebäude, ein Platz, ein Museum o.Ä., das besonders schön oder interessant ist, so dass es Touristen oft besichtigen“ [Fp].

An diesen Definitionen fällt zum einen auf, dass der Begriff in beiden Beispielen in Verbindung mit dem Tourismus steht. Zum anderen wird eine Sehenswürdigkeit als etwas besonderes beschrieben. Diese Besonderheit kann die verschiedensten Ausprägungen annehmen. In dem Unterkapitel 2.2 wird dieser sehr abstrakte Begriff genauer charakterisiert.

2.2. Städtetourismus

Eine Sehenswürdigkeit ist ein zentrales Element des Tourismus'. Bestimmte Facetten des Tourismus' sind einzig auf das Besichtigen dieser interessanten Objekte ausgelegt. Da in dieser Arbeit eine Stadt als Beispiel verwendet wird. lohnt sich ein kurzer Blick auf eine bestimmte Facette namens Städtetourismus. Da dieses Thema jedoch nicht Hauptbestandteil der Arbeit ist, soll hier lediglich ein kurzer Überblick vermittelt werden.

2.2.1. Definition und Beschreibung

Genau wie bei den Sehenswürdigkeiten, so fällt auch die Definition des Städtetourismus' eher abstrakt aus. So beschreiben Claudia Anton und Heinz-Dieter Quack, dass der Städtetourismus nicht genau eingeordnet werden kann. Grund dafür sind „verschiedene Ursachen [...] und Ausprägungen" [LS05, S. 9]. Sie sehen den Städtetourismus eher als Oberbegriff [LS05, S. 9]. Der Klettverlag rückt in seiner Definition die „"Erkundung und das „kulturelle Angebot" [Lex07] in den Vordergrund. In 2.2.2 werden jedoch noch andere Ausprägungen aufgeführt. Die Ursachen erklären sich die Autoren durch immer neu auftretende Marktstrukturen, die die Menschen in die Städte ziehen [LS05, S. 7].

Die gelben Punkte in Abbildung 2.1 zeigen Städte in Deutschland mit Besichtigungstourismus. Man kann daher annehmen, dass an diesen Standorten viele Sehenswürdigkeiten existieren. Unter diesen Städten wird auch Köln aufgeführt.

2.2.2. Verschiedene Arten des Städtetourismus

Wie bereits festgestellt, gibt es viele verschiedene Ausprägungen des Städtetourismus'. In diesem Abschnitt sollen kurz, ohne Garantie auf Vollständigkeit, einige Ausprägungsbeispiele aufgeführt werden. In ihrem Buch „Geographie der Freizeit und des Tourismus" führen die Autoren Becker et al. verschiedene Arten des Tourismus' vor, bei denen es sehr wahrscheinlich ist, dass der Reisende eine Stadt besucht. [BHS03, S. X f.]:

- Kongress- und Tagungstourismus
- Industrietourismus
- Gesundheitstourismus
- Tagungstourismus

Als weitere Beispiele nennen Claudia Anton und Heinz-Dieter Quack [LS05, S. 7]:

Abbildung 2.1.: Fremdenverkehrsgebiete und Städtetourismus in Deutschland [Lat02, S. 165]

- Shopping
- Musicals
- Events
- Bildung

Das Kategorisieren des Städtetourismus' ist in soweit wichtig, da sich mit den verschiedenen Interessen der Menschen auch die Schwerpunkte der Sehenswürdigkeiten ändern. In 2.3 sollen einige dieser Schwerpunkte am Beispiel der Stadt Köln aufgeführt werden, bevor in 2.4 eine abschließende Begrenzung des Begriffes einer Sehenswürdigkeit für diese Arbeit getroffen wird. Dadurch muss sich nicht auf alle Eventualitäten konzentriert werden.

2.3. Beispiel Köln

Als eine der vier Millionenstädte in Deutschland bietet Köln eine große Menge verschiedener Sehenswürdigkeiten an, die die unterschiedlichsten Menschen anziehen. Eine kleine nicht repräsentative Umfrage, die im Rahmen dieser Arbeit durchgeführt wurde, ergab, dass 66,6% der befragten Personen eine andere Vorstellung zur zweitwichtigsten Sehenswürdigkeit in Köln hatten.[1]

Die Internetseite `www.koelntourismus.de` stellt ein großes Angebot von Attraktionen der Stadt Köln vor. Die einzelnen Attraktionen lassen sich grob in die folgenden Kategorien einteilen:

- **Veranstaltungen und Events**: Darunter zählen unter anderem Messen, die Kölner Lichter und der Karneval.
- **Unterhaltung und Kommerz**: Darunter zählen unter anderem Shopping, Gastronomie und Brauhäuser, Rheinaktivitäten, Sport und Kinos.
- **Kulturell und Historisch**: Darunter zählen unter anderem Gotteshäuser, Opern und Theater, Gebäude und Naturgebiete.
- **Bildung und Gesundheit**: Darunter zählen unter anderem Hochschulen und Unis, Krankenhäuser und Fachärzte und Bibliotheken.

[1] vgl. Anhang A

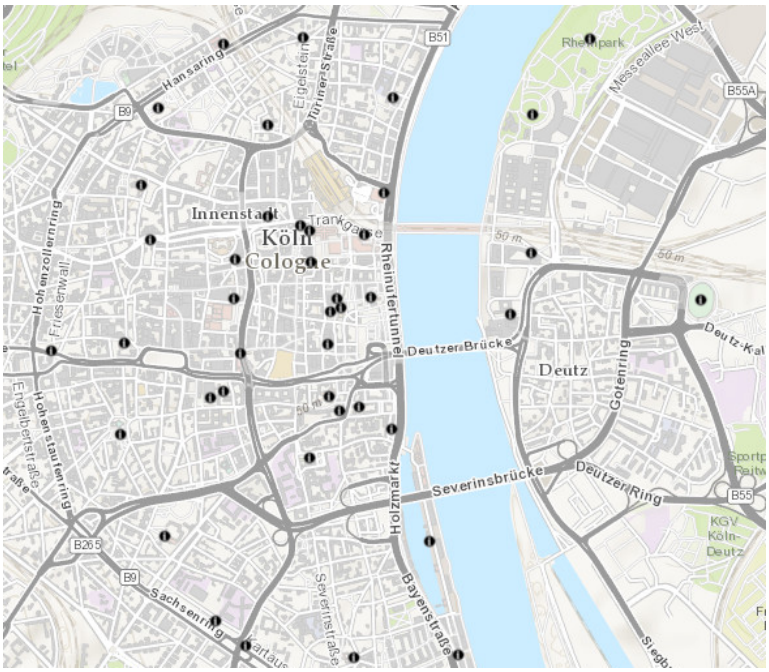

Abbildung 2.2.: Ausschnitt aus „Köln - Sehenswürdigkeiten“[Red16a]

Jede dieser Kategorien bedient eine eigene Zielgruppe von Menschen und jede dieser Kategorien bietet besondere Sehenswürdigkeiten, die in der Stadt verteilt sind. So ist es beispielsweise weniger wahrscheinlich, dass sich Unterhaltungs- und Kommerztouristen - mit Ausnahme des Kölner Doms - die Kirchen der Stadt angucken werden.

Die Stadt Köln bietet unter anderem als Teil der Open Gouverment Data Bewegung einen Datensatz an, der einige Sehenswürdigkeiten, wenn auch längst nicht alle, beinhaltet [Red16b]. Nach welchen Kriterien diese Daten ausgewählt wurden ist nicht angegeben. Die Abbildung 2.2 zeigt dabei einen Ausschnitt einer Karte von Köln. Die Sehenswürdigkeiten, die in dem Datensatz vorhanden sind, werden durch die schwarzen Punkte dargestellt.

2.4. Fokus dieser Arbeit

In den vorherigen Abschnitten 2.1 bis 2.3 wurde das große Ausmaß beschrieben, die Sehenswürdigkeiten mit sich bringen. Es wurde festgestellt, dass die Stadt Köln sehr viele Sehenswürdigkeiten besitzt, die jeweils andere Zielgruppen ansprechen. In dieser Bachelorarbeit sollen Sehenswürdigkeiten der Stadt ermittelt werden. Aufgrund der vielen Zielgruppen müssten jedoch auch eine Menge von Bedürfnissen und Erwartungen dieser Gruppen erarbeitet werden. Dieser Aufwand würde vom eigentlichen Ziel,

welches in 1.2 beschrieben wurde, zu sehr ablenken. Daher soll sich in dieser Arbeit auf einen kleineren Teil konzentriert werden, der in diesem Abschnitt festgelegt wird.

Die in 2.3 beschriebenen Kategorien eigenen sich gut, um den Fokus auf bestimmte Arten von Sehenswürdigkeiten zu konzentrieren. Diese Arbeit spezialisiert sich auf die Kategorie **Kulturell und Historisch**. Die Stadt Köln hat eine sehr lange Geschichte und daher sollte es angebracht sein, sich auf die historischen Merkmale zu fokussieren. Kommerzielle Veranstaltungsgebäude wie Opern oder Theater sollen ebenfalls vernachlässigt werden. Des Weiteren wird die Zeitspanne eingeschränkt. Es werden nur die Sehenswürdigkeiten berücksichtigt, die aus der Zeit des römischen Reiches, dem Mittelalter und der frühen Neuzeit stammen.

3. Linked Open Data

Bisher wurden in Kapitel 2 wesentliche Grundlagen einer Sehenswürdigkeit für diese Arbeit erarbeitet und diese auf die Stadt Köln bezogen. In diesem Kapitel werden die technischen Grundlagen dieser Arbeit erläutert. Die Abschnitte 3.1 und 3.2 stellen den Begriff und die Verwendung von Linked Open Data genauer vor. Es folgt in 3.3 eine Beschreibung des Semantic Web, welches in seinen Teilen und dessen Funktionsweise kurz erklärt wird. Kapitel 4 stützt sich auf diese Grundlagen. Abschließend wird die Linked Open Data Cloud in 3.4 beschrieben.

3.1. Begriffserklärung

Unter Linked Open Data, zum Teil auch nur Linked Data genannt, versteht man die Erfolgsmethode, um strukturierte Daten von verschiedenen Datenquellen miteinander zu verknüpfen und zu veröffentlichen. [BBDR+13, S. 599] [BHBL09]. Geprägt wurde der Begriff unter anderem stark von Tim Berners-Lee. In seinem Artikel „Linked Data - The Story So Far“ schildert er die Maxime, der Linked Data zu Grunde liegt. Diese besagt, dass die im Web veröffentlichten Daten in einem Format vorliegen müssen, sodass:

1. diese von Maschinen verarbeitet werden können,
2. der Inhalt der Daten genau definiert ist,
3. die Daten mit anderen externen Daten verbunden werden können und
4. die Daten von anderen externen Daten verbunden werden können [BHBL09].

Als Technologie, die diese Kriterien erfüllt, benennt Berners-Lee im gleichen Artikel das Resource Description Framework (RDF), welches in 3.3.2 vorgestellt wird. Linked Data ist somit keine eigene technologische Erfindung, sondern viel mehr das Konzept, dass alle Daten im Semantic Web beschreibt, die durch das RDF erstellt wurden. Das Semantic Web, welches in 3.3 im Detail beschrieben wird, existiert aufgrund der Verwendung von Linked Data [BHBL09] [BL09].

Der Zusatz *Open* in Linked Data bedeutet, dass die Informationen der Daten und die Daten selbst unter einer Lizenz stehen müssen, die es jedem erlaubt, diese Daten weiterzuverwenden [BL09]. Es steht auch allen frei neue Daten zu erstellen und diese im Semantic Web zu veröffentlichen.

3.2. Prinzipien von Linked Open Data

Im Jahr 2006 stellte Berners-Lee vier Richtlinien vor, wie Daten im Semantic Web zu veröffentlichen und zu verbinden seien.

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

[BL09]

Zwar betont er, dass das Nichteinhalten dieser Regeln nichts zerstöre, jedoch könne nur so sichergestellt sein, dass Daten untereinander verbunden werden [BL09].
Anhand dieser Regeln, die heute als „Linked Data Principles“ [BHBL09] bekannt sind, werden auch die vier Basiskomponenten von Linked Data deutlich:

1. **Uniform Resource Identifier (URI)** Dient zur eineindeutigen Identifikation von Ressourcen und Daten.
2. **Hypertext Transfer Protocol (HTTP)** Das Übertragungsprotokoll, welches auch für das „web of hypertext“ [BL09] benutzt wird.
3. **RDF** Dient als Datenmodell für Linked Data.
4. **SPARQL** Abfragesprache für Linked Data.

Sowohl RDF, als auch SPARQL werden im Abschnitt 3.3 und dessen Unterabschnitte genauer behandelt.

3.3. Semantic Web

3.3.1. Einführung

3.3.1.1. Name und Synonyme

Das Semantic Web ist in der Literatur unter einigen verschiedenen Namen bekannt. Daher ist es wichtig zu wissen, dass die verschiedenen Begriffe das Gleiche bedeuten. Die folgenden Namen werden dominant verwendet:

- **Semantic Web**: Erstmalig von Tim Berners-Lee im Jahr 2000 verwendet und beschrieben [BL00, S. 191].
- **Web of Data**: Gleichwertiger Begriff, der unter anderem auch von Berners-Lee verwendet wird [BHBL09].
- **Web 3.0**: Wurde in Anlehnung an Web 2.0 von John Markoff erfunden [Mar06].

3.3.1.2. Definition

Die Ursprungsidee des Semantic Web war es vor allem dem Problem entgegenzugehen, dass Daten im Internet roh vorhanden sind. Dadurch werden nach Bizer et al. viele Strukturen und Semantiken aufgegeben [BHBL09]. Eine präzise, kurze Definition liefern Berners-Lee et al. in „The Semantic Web“ und beschreiben somit wie das Problem gelöst werden soll:

> The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [BLHL+01, S. 3 f.]

Nachfolgend sollen aus dieser Definition noch einmal die wichtigsten Attribute herausgeschrieben werden:

1. Das Semantic Web ist kein eigenständiger Raum, sondern Teil des Internets.
2. Informationen - also Daten - sollen eine definierte Struktur besitzen.
3. Die Daten sind sowohl für den Menschen, als auch für den Computer verwertbar.

Guha et al. beschreiben das Semantic Web nicht als Web der Dokumente, sondern eher als Web der Beziehungen zwischen Ressourcen, die die reale Welt abbilden [GMM03, S. 700]. Was genau unter den Ressourcen zu verstehen ist, wird in 3.3.2 beschrieben.

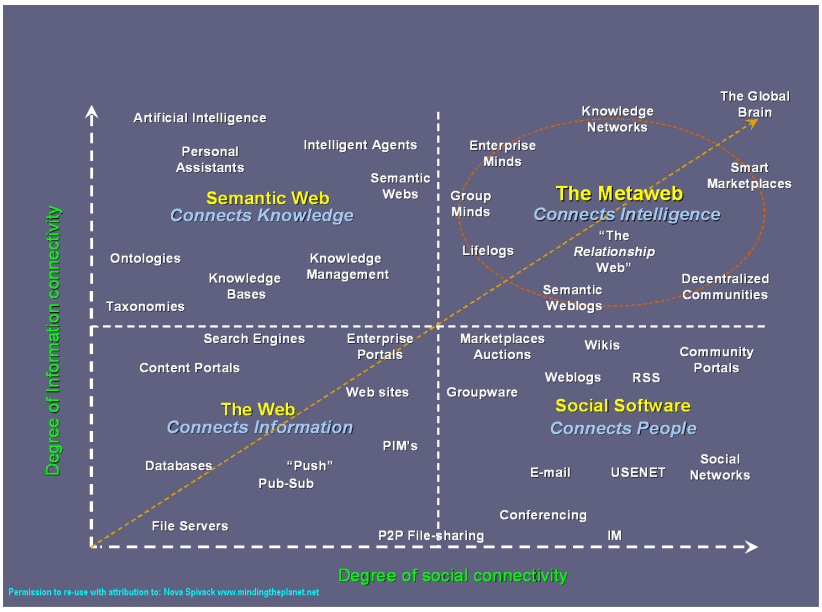

Abbildung 3.1.: Verhältnis von Semantic Web und Social Media [Spi04]

3.3.1.3. Abgrenzung vom alltäglichen Internet

Internetseiten und -dienste werden heutzutage von jedem fast täglich verwendet. Die Menschen sind in der Lage sich über diese Plattformen gegenseitig auszutauschen und soziale Kontakte zu pflegen. Häufig fallen hier Begriffe wie Web 2.0 [O'r07] oder aktueller *Social Media* [Sch10]. In ihrem kurzen Artikel „Linked Data" treffen Gradmann et al. die Aussage, dass die Dokumente und der Inhalt in diesem Teil des Internets nur für Menschen bestimmt seien, da auch nur diese den impliziten Kontext verstehen [GHO12]. Die sozialen Aspekte stehen im Semantic Web, wie Abbildung 3.1 zeigt, nicht im Fokus.

3.3.1.4. Technologien

Viele Begriffe und Konzepte des Semantic Web haben ihren Ursprung aus der Forschung der künstlichen Intelligenz und existierten bereits bevor das Internet entwickelt wurde [BLHL+01, S. 5]. Die Abbildung 3.2 zeigt die Komponenten, die das Semantic Web bilden. In den Abschnitten 3.3.2 bis 3.3.4 werden die technische Komponenten vorgestellt, die speziell für das Semantic Web existieren. Darunter zählen zum einem das Resource Description Framework und der Begriff der Ontologie und zum anderem die Abfragesprache SPARQL. Diese basieren, wie Abbildung 3.2 zeigt, zum Teil auf anderen bekannten Kerntechnologien der Informatik wie URI, UNICODE oder XML.

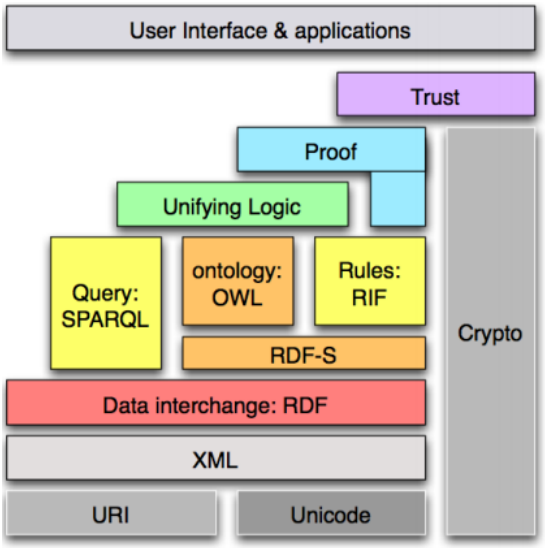

Abbildung 3.2.: Aufbau des Semantic Web [Bra06, S. 24]

3.3.2. Resource Description Framework

Das RDF ist ein graphenbasiertes Datenmodell, womit strukturierte Daten, auch Metadaten genannt, verlinkt, ausgetauscht und verarbeitet werden können [BHBL09] [Mil98, S. 15].

3.3.2.1. Ursprung

Seinen Ursprung hatte das RDF 1995 mit dem Webstandard Platform for Internet Content Selection (PICS). Mit Hilfe von PICS konnten Webseiten durch das Platzieren von Meta-Tags bewertet werden [o.V09]. Als Anwendungsbeispiel führt Miller auf, dass durch das Setzen von diesen Bewertungen, Eltern eigene Filter erstellen konnten, um unerwünschte Webseiten für ihre Kinder zu blockieren [Mil98, S. 16]. Auf Grundlage dieser vielseitig einsetzbaren Idee entstand 1997 vom World Wide Web Consortium (W3C) der erste Entwurf vom RDF. Später im Jahr 1999 wurde das RDF spezifiziert, indem dessen Modell und Syntax beschrieben wurden [Las99]. Aktuell existiert das RDF in der Version 1.1.

3.3.2.2. Datenmodellierung

1998 erklärte Eric Miller in seinem Artikel „An Introduction to the Resource Description Framework“ den Aufbau von Ressourcen, die mit dem RDF beschrieben werden können. Er charakterisiert eine Ressource als ein, durch eine URI eineindeutig identifizierbares semantisches Objekt, dass bestimmte Attribute (engl.: properties) besitzt.

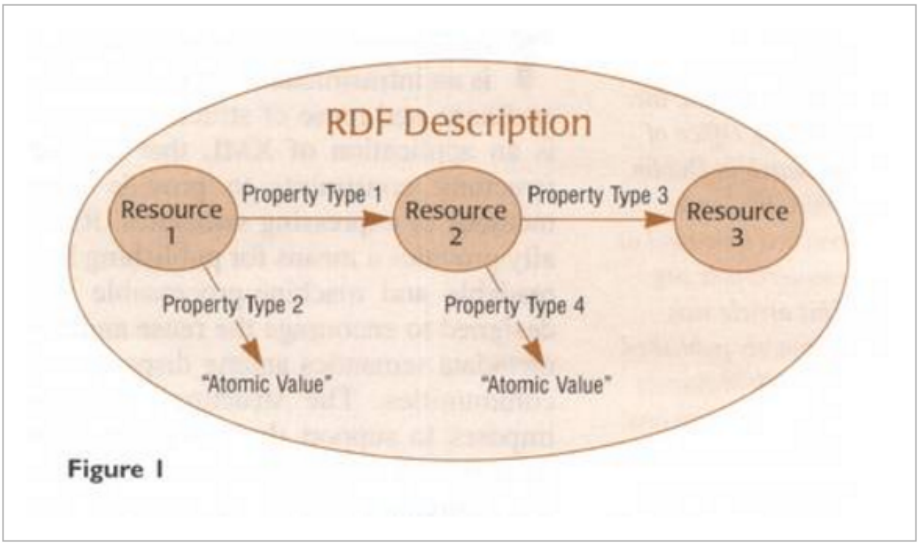

Abbildung 3.3.: RDF Beschreibung nach Miller [Mil98, S. 16]

Attribute können je nach Typ Werte annehmen, die atomar oder ihrerseits wieder Ressourcen sind. Miller definiert in diesem Zusammenhang auch der Begriff einer RDF Beschreibung als eine Sammlung von Eigenschaften, die auf die gleiche Ressource verweist. [Mil98, S. 16]. Abbildung 3.3 zeigt eine solche RDF Beschreibung.
Im RDF Modell werden Aussagen über Ressourcen, wie sie nach Miller vorgestellt wurden, getroffen. Eine Aussage wird durch ein Tupel aus drei Elementen, auch Tripel genannt, codiert [Las99]. Die drei Elemente werden in der Literatur auch als *Subjekt*, *Prädikat*, und *Objekt* bezeichnet [BHBL09]. Das Subjekt wird durch das Prädikat mit dem Objekt in Beziehung gebracht. Es entsteht ein gerichteter Graph, der auch in Abbildung 3.3 an den Pfeilen zu erkennen ist. [Las99]. Sowohl das Subjekt, als auch das Prädikat sind immer Ressourcen. Das Objekt ist mit dem von Miller vorgestellten Attributwerten vergleichbar und kann sowohl eine Ressource, als auch ein atomarer Wert sein.

3.3.2.3. Syntax

Für RDF Modelle sind verschiedene Darstellungsformate möglich. Das W3C stellt dazu unter `https://www.w3.org/standards/techs/rdf` einige Standards zur Verfügung. Darunter gehören unter anderem:

- **N-Triples** Sehr einfache Darstellung, bei welcher die Elemente eines Tripels zeilenweise, getrennt durch Leerzeichen, aufgeschrieben werden. Ein Punkt am Ende der Zeile beendet die Aussage [Bec14].

- **RDF/XML** Sehr verbreitete und, wie anhand von Abbildung 3.2 zu erahnen, auch die empfohlene Darstellungsform. Hier liegt die XML Syntax zugrunde. Miller sieht viele Vorteile bei der XML Syntax. Sie bietet eine Anbieterunabhängigkeit, Erweiterbarkeit, Validierbarkeit, Lesbarkeit durch einen Menschen und die Möglichkeit komplexe Strukturen darzustellen [Mil98, S. 15].

Allen Darstellungsformen liegt jedoch der Vorteil vor, dass durch die strukturierte Codierung sowohl Menschen, als auch Maschinen die gespeicherten Metadaten verarbeiten können. Bei der XML Syntax wird diese Eingenschaft durch die Verwendung von XML Namespaces erreicht. Des Weiteren können die URIs der Tripel Elemente beliebig gewählt werden. Dadurch unterstützt RDF eine modulare Interoperabilität zwischen einzelnen Datensätzen.

Listing 3.1: RDF/XML Dokument [o.Vb]

```
1  <?xml version="1.0"?>
2
3  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       ↪ xmlns:si="http://www.w3schools.com/rdf/">
4
5  <rdf:Description rdf:about="http://www.w3schools.com">
6  <si:title>W3Schools</si:title>
7  <si:author>Jan Egil Refsnes</si:author>
8  </rdf:Description>
9
10 </rdf:RDF>
```

3.3.3. Ontologien

Ontologien sind nach Berners-Lee eine tragende Komponente des Semantic Web und beschreiben Sammlungen von Informationen [BLHL+01, S. 9]. In Abbildung 3.2 sind Ontologien ebenfalls mit der Ergänzung „OWL“ aufgeführt. Der Ursprung des Begriffs kommt aus der Philosophie. Es ist eine Theorie des Seienden, dessen Möglichkeiten und Bedingungen [Hes02]. In der Informatik, besonders bei der künstlichen Intelligenz und im Web, so Berners-Lee, sind Ontologien Dokumente oder Dateien, die Beziehungen unter Begrifflichkeiten, sogenannten Vokabeln, formal definieren. Innerhalb einer Ontologie kann eine Taxonomie, also eine Klassifizierung, enthalten sein und bestimmte Regeln definiert werden [BLHL+01, S. 9].
Dostal el al. machen darauf aufmerksam, dass es praktisch nicht umsetzbar ist, ein allgemeingültiges Vokabular zu schaffen [DJK04]. Verschiedene Domänen können den

gleichen Begriff in einem unterschiedlichen Kontext benutzen. Jedoch ist es auch möglich, dass verschiedene Domänen zwei unterschiedliche Vokabeln benutzen, beide meinen jedoch das Gleiche. Das W3C hat diese Problematik erkannt und entwickelte darauf hin die Web Ontology Language (OWL). Mittels OWL können einerseits verschiedene Vokabeln abgeglichen werden, andererseits bietet sie auch die Möglichkeit neue Ontologien formal zu erstellen. OWL ist dabei im eigentlichen Sinne keine Sprache, wie es bei Java oder XML zu erwarten wäre, sondern ist ein RDF Graph bestehend aus RDF Tripeln. Mittels Resource Description Framework Schemas (RDFSs) und OWL werden Vokabeln erstellt um Entitäten oder deren Beziehungen zu beschreiben [BHBL09].
Das W3C weist auf der eigenen Webseite darauf hin, dass die Begriffe Vokabeln und Ontologie nicht eindeutig voneinander zu unterscheiden sind und synonym verwendet werden können [o.V15].
RDFSs erlauben, ähnlich wie XML Schemas, das Deklarieren von Vokabeln und das Definieren gültiger Attributwerte. Durch Namespaces werden RDFs eineindeutig identifiziert.

3.3.4. SPARQL

In den beiden vorherigen Abschnitten 3.3.2 und 3.3.3 wurde erklärt wie die strukturierten Daten im Semantic Web vorliegen. In diesem Abschnitt wird die Technologie vorgestellt, mit der diese Daten abgefragt werden können.

SPARQL ist eine graphenbasierte Abfragesprache für das RDF-Datenmodell und Kerntechnologie im Semantic Web, wie Abbildung 3.2 zeigt. Das Akronym steht dabei für SPARQL Protocol and RDF Query Language [Bec11]. Entwickelt und standardisiert wurde die Sprache von der RDF Data Access Working Group des W3C. Dadurch ist SPARQL die de-facto Standardabfragesprache im Semantic Web.
In den folgenden Abschnitten wird die Sprache genauer vorgestellt.[1] Nach Berners-Lee sollte jeder große Datensatz einen SPARQL Abfragedienst haben [BL09].

3.3.4.1. Abfrage

Die Autoren Perez, Arenas und Gutierrez unterteilen eine SPARQL Abfrage in drei Teile [PAG06, S. 30]:

1. **Pattern Matching** In diesem Teil wird die vorher vom Nutzer spezifizierte Datenquelle nach einem Muster befragt. Hier sind verschiedene Operationen ver-

[1] Eine vollständige Spezifikation ist auf der Webseite des W3C unter `https://www.w3.org/TR/rdf-sparql-query/` zu finden.

fügbar. Darunter auch einige Operationen wie `UNION`, die aus der relationalen Algebra bekannt sind.[2]

2. **Solution Modifiers** Bei diesem Teil kann das Ergebnis der Abfrage modifiziert werden. Möglich sind, wie die Autoren Perez et al. es nennen, die klassischen Operatoren. Darunter nennen sie beispielsweise das Sortieren, das Limitieren oder Eliminieren von Duplikaten [PAG06, S. 30].

3. **Output** Dieser Teil beschreibt das Abfrageergebnis und kann aus verschiedenen Formen bestehen. So sind unter anderem binäre Antworten, als auch die Konstruktion neuer Tripel aus den Ergebnissen möglich.

3.3.4.2. Syntax

In diesem Abschnitt werden kurz die wichtigsten Syntaxmerkmale von SPARQL anhand des Codesbeispiels 3.2 erklärt. Die Abfrage soll alle Musikartisten mit Namen, Bild, Link zur Homepage und ihrem Ort finden. Die Ausgabe wird auf zehn Treffer limitiert.

Durch den `PREFIX` Operator in den Zeilen eins und zwei aus Listing 3.2 können verschiedene Quellen von Ontologien außerhalb des eigentlichen Abfrageblocks spezifiziert werden. Vergleichbar ist diese Eigenschaft mit den XML Namespaces. Der Vorteil ist, dass der vollständige URI durch das definierte Präfix abgekürzt werden kann, was den Abfrageteil leserlicher gestaltet.

Der Operator `WHERE` definiert den Abfrageblock. Hier können Variablen, gekennzeichnet mit einem führendem `?` oder `$`, definiert werden.
In diesem Beispiel werden alle RDF Tripel zusammengefasst, dessen Subjekt vom Typ `no:MusicArtist` ist. Die Übereinstimmung wird durch das Prädikat `a` überprüft.[3] Zudem müssen alle gefundenen Subjekte über die Eigenschaften `foaf:name`, `foaf:img`, `foaf:homepage` und `foaf:based_near` verfügen. Die Werte werden in die entsprechenden Variablen, die in dieser Zeile stehen gespeichert. Diese Abfrage erfolgt in den Zeilen fünf bis neun des Listing 3.2.
Innerhalb eines `WHERE` Blocks können auch weitere `SELECT` Ausdrücke existieren. Hinter einem `WHERE` können Solution Modifier Operatoren angefügt werden. In dem Listing 3.2

[2] Obwohl einige Operatoren sowohl in SPARQL, als auch beispielsweise SQL unter dem gleichen Begriff vorkommen, soll an dieser Stelle bemerkt werden, dass sich die beiden Sprachen durchaus in einigen Operatoren unterscheiden können.
[3] Das Prädikat `a` ist eine Abkürzung für `rdf:type`

wird das Ergebnis in Zeile 10 auf zehn Eintrage limitiert.

Der Operator `SELECT` definiert die Variablen, die beim Ergebnis projiziert werden sollen.

Zeilenkommentare beginnen mit einem führendem # Zeichen.

Das Codebeispiel 3.3 ist semantisch identisch zu 3.2. Anstelle der Semikolons wurden in 3.3 Punkte verwendet. Ein Semikolon am Ende einer Zeile bedeutet, dass das Subjekt in der nächsten Zeile weiter verwendet wird, während es beim Punkt immer neu definiert werden muss. Diese Art von Syntax wird vom W3C als Turtle bezeichnet [BBLPC14].

Listing 3.2: Abgewandelter SPARQL Beispielcode nach Feigenbaum [Fei09]

```
1  PREFIX mo: <http://purl.org/ontology/mo/>
2  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3  SELECT ?name ?img ?hp ?loc
4  WHERE {
5    ?artist a mo:MusicArtist ;
6    foaf:name ?name ;
7    foaf:img ?img ;
8    foaf:homepage ?hp ;
9    foaf:based_near ?loc .
10 } LIMIT 10
```

Listing 3.3: Semantisch gleicher Code wie in Listing 3.2

```
1  PREFIX mo: <http://purl.org/ontology/mo/>
2  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3  SELECT ?name ?img ?hp ?loc
4  WHERE {
5    ?artist a mo:MusicArtist .
6    ?artist foaf:name ?name .
7    ?artist foaf:img ?img .
8    ?artist foaf:homepage ?hp .
9    ?artist foaf:based_near ?loc .
10 } LIMIT 10
```

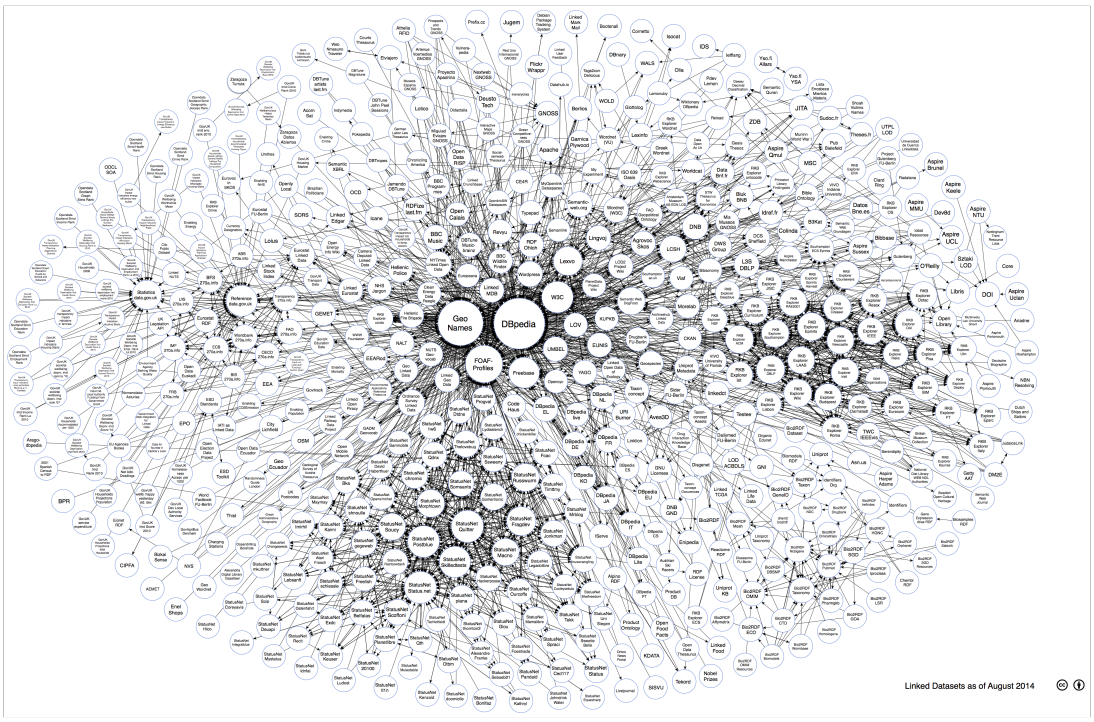

Abbildung 3.4.: Datenbestände und deren Beziehungen in der LOD Cloud [CJ14]

3.4. Linked Open Data Cloud

„Die Linked Open Data Cloud bildet alle Datensätze ab, die im Web of Data im Linked Data Format veröffentlicht wurden [GHO12, S. 20].“ Abbildung 3.4 zeigt, welche großen Datensätze im Jahr 2014 existierten. Anhand der Größe der Kreise, wird die Anzahl der RDF Tripel in diesem Datensatz dargestellt. Welche Relationen einzelne Datensätze untereinander haben, zeigen die Pfeile. Deutlich zu erkennen ist, dass *DBpedia* und *GeoNames* die größten Datenbestände haben. Des Weiteren referenzieren fast alle anderen Datensätze auf DBpedia oder GeoNames. Die Linked Open Data Cloud ist frei für jeden verfügbar. Daten in der Cloud können bei den meisten Datensatzanbietern über mehrere Methoden abgerufen werden. Im Beispiel von DBPedia stehen die folgenden Alternativen zu Wahl [BLK+09]:

1. Zugriff per URI: Die Daten können direkt im Browser über einen URI angefragt werden. Die Darstellung der erhaltenen Daten kann verschieden sein.

2. Zugriff per SPARQL Endpunkt: Die Daten können mittels eines SPARQL Query abgefragt werden. Der Datensatzanbieter stellt dazu einen SPARQL Endpunkt bereit. Die Darstellung der erhaltenen Daten kann verschieden sein.

3. Zugriff per RDF Dumps: Die Daten können direkt als RDF Datensatz heruntergeladen werden. Je nach Größe des Datensatzes können mehrere Gigabytes Speicher benötigt werden. Die Darstellung der erhaltenen Daten kann verschieden sein, häufig wird jedoch RDF/XML angeboten.

4. Zugriff per Schlüsselwortsuche: Anhand von Schlüsselwörtern können RDF Beschreibungen im Datensatz gesucht werden. Die Darstellung der erhaltenen Daten kann verschieden sein.

Bauer et al. gaben 2011 einen Datenbestand von über 31 Milliarden RDF Tripel und in etwa 504 Millionen RDF Links innerhalb der Linked Open Data Cloud an [BK11, S. 40s].

4. Entwicklung einer SPARQL Abfrage

In diesem Kapitel soll eine SPARQL Abfrage herausgearbeitet werden, welche Sehenswürdigkeiten der Stadt Köln ermittelt. In dem Abschnitt 2.4 wurden bereits die Kriterien definiert, nach der Sehenswürdigkeiten gesucht werden sollen. An dieser Stelle sollen sie noch einmal stichpunktartig aufgelistet werden:

Eine relevante Sehenswürdigkeit für diese Arbeit

- befindet sich im Stadtgebiet von Köln,
- ist für kulturell und historisch versierte Touristen von Bedeutung,
- ist kein kommerzielles Veranstaltungsgebäude wie etwa eine Oper oder Theater und
- hat ihren Ursprung aus der Römerzeit, dem Mittelalter oder der frühen Neuzeit.

4.1. Vorgehensweise

In diesem Abschnitt werden in 4.1.1 kurz die Technologien benannt, mit denen die Abfrage erfolgt. Anschließend werden in 4.1.2 auf Basis von Ramanathan Guha et al. [GMM03] zwei Suchprinzipien vorgestellt, die sich im Semantic Web anwenden lassen. Zum Abschluss wird kurz eine Abgrenzung zu der hier vorgestellten Methode und den verfügbaren Suchmaschinen erfolgen.

Im nächsten Schritt werden in 4.2 potentielle Datensatzquellen identifiziert und bewertet. Aus den gefundenen Datensätzen sollen letztlich die Sehenswürdigkeiten ermittelt werden.

Bevor in 4.5 die erarbeitete Abfrage mit ihrem Ergebnis vorgestellt wird, wird der Entwicklungsprozess in 4.3 und 4.4 genauer beschrieben. Wichtige Beobachtungen und Erkenntnisse beschreibt der Abschnitt 4.6.

4.1.1. Technologien

In Kapitel 3 wurde der Begriff Linked Open Data und das Semantic Web genauer erklärt.
Für diese Arbeit ist die Abfragesprache SPARQL besonders relevant. Zwar wurden in 3.4 mehrere Möglichkeiten vorgestellt, wie auf Daten in einem Datensatz zugegriffen werden kann, doch SPARQL bietet, dank seiner vielseitigen Möglichkeiten eine Abfrage zu gestalten, eine gute Grundlage. SPARQL hat sich als Basiskomponente des Semantic Web etabliert. Das garantiert, dass viele, besonders die großen Datensatzanbieter einen SPARQL Endpunkt besitzen, auf den zugegriffen werden kann.
Neben SPARQL werden auch die Ontologien der Datensätze eine ganz wichtige Rolle spielen, da durch diese eine Suchabfrage erst genau spezifiziert werden kann.
Das RDF stellt die Grundlage des Semantic Web dar, jedoch ist es für diese Arbeit nicht von zentraler Bedeutung. Zwar muss auch diese Technologie verstanden werden, da jedoch die Abfrageergebnisse nicht weiter verarbeitet oder neue Datensätze erstellt werden sollen, rücken besonders die Darstellungsformen wie RDF/XML in den Hintergrund. Wichtig ist viel mehr das Verständnis der RDF Tripel.

4.1.2. Suchprinzipien

Nach Aussage von Guha, McCool und Miller ist das Suchen im Internet die beliebteste Anwendung im Internet. Gleichzeitig sehen sie in diesem Bereich jedoch auch ein großes Verbesserungspotential. Die Autoren sind der Meinung, dass durch das Suchen im Semantic Web, Suchergebnisse im Web der Dokumente verbessert werden können. Die Suche im Semantic Web klassifizieren sie dabei in zwei Arten [GMM03, S. 702]:

1. **Navigierende Suche:** Hier gibt der Nutzer Wörter in eine Suchmaschine ein, die er in einem Dokument im Internet finden möchte. Die Suchmaschine dient als Navigationstool der gefundenen Webseiten.

   Bsp.: Der Nutzer sucht ein bestimmtes Video, weiß den Titel aber nicht mehr vollständig. Er gibt die Wörter, die er noch kennt ein und navigiert sich durch die Suchergebnisse zum passenden Video.

2. **Erforschende Suche:** Hier gibt der Nutzer Wörter in eine Suchmaschine ein, die eine reale Sache beschreiben, um Informationen darüber zu erhalten. Der Nutzer weiß nicht, welche Antworten er bekommt.

   Bsp.: Der Nutzer hat kürzlich einen Film mit einem ihm unbekannten Schauspieler gesehen. Er sucht nun nach diesem Namen, in der Hoffnung Informationen wie weitere Werke oder Nationalität zu erfahren.

Auf welche Weise eine Suchanfrage gestellt wird, ist von der Intention des Nutzers abhängig. Wird nun das Thema dieser Arbeit, die Ermittlung von Sehenswürdigkeiten, auf die vorgestellten Suchprinzipien projiziert, so wird deutlich, dass für die Entwicklung einer Abfrage beide Varianten möglich sind. Sind die zu ermittelnden Sehenswürdigkeiten bereits bekannt, würde eine navigierende Suche vorliegen. In dieser Arbeit soll jedoch der komplexere Weg betrachtet werden:
Die zu ermittelnden Sehenswürdigkeiten sind nicht oder nur teilweise bekannt. Dass der Kölner Dom eine Sehenswürdigkeit ist, die die oben genannten Kriterien erfüllt sollte offensichtlich sein, jedoch soll in der zu entwickelnden Abfrage der Kölner Dom nicht explizit angegeben werden. In diesem Falle soll auch angenommen werden, dass der Kölner Dom nicht bekannt ist. Durch diese Bedingung kann zudem festgestellt werden, ob eine erforschende Suche im Semantic Web überhaupt zu befriedigenden Ergebnissen führt.

4.1.3. Abgrenzung zu Suchmaschinen

In diesem Abschnitt werden kurz einige Unterschiede zwischen einer traditionellen Suche im Web der Dokumente und einer Suche im Semantic Web aufgeführt.
Gradmann et al. betonen, dass traditionelle Suchmaschinen die Semantik und den Kontext von den gefundenen Dokumenten nicht verstehen [GHO12, S. 18]. Das Semantic Web ist jedoch dafür ausgelegt, dass zu einer Ressource auch weitere semantische Verknüpfungen vorhanden sind. Semantic Web Suchmaschinen hingegen können Linked Data über URIs zwischen verschiedenen Datenquellen erforschen.[BHBL09] [Hau09, S. 71]. Des Weiteren bemerken Bizer und Berners-Lee, dass Linked Data Anwendungen auf einem ungebundenen globalen Datenraum arbeiten. Die Antworten sind daher vollständiger [BHBL09].

Suchmaschinenanbieter wie Google haben sich in den vergangenen Jahren stark weiterentwickelt. Hier kann von einer traditionellen Suchmaschine nicht mehr die Rede sein. Die Abbildung 1.1 zeigt wie Google einerseits den Suchbegriff „sehenswürdigkeiten köln“ als „Köln / Interessante Orte“ interpretiert. Andererseits erhält der Nutzer sowohl Treffer zur Stichwortsuche, die das traditionelle Element von Google bildet, sowie weitere Informationen zur Stadt Köln. Google erweitert somit die Suchergebnisse um semantische Elemente. Wie genau diese Informationen ermittelt werden ist nicht bekannt, jedoch geht aus einem Artikel des *Wall Street Journals* heraus, dass Google eigene Methoden entwickelt hat [Efr12].

4.2. Vorhandene Datensatzquellen

Im Abschnitt 3.4 wurde bereits die Linked Open Data Cloud beschrieben. In diesem Abschnitt sollen Datensatzquellen vorgestellt und genauer betrachtet werden, die für die Ermittlung der Sehenswürdigkeiten in Betracht kommen.

4.2.1. Auswahlkriterien

Die Linked Open Data Cloud ist mit über 31 Milliarden RDF Tripel sehr groß. Wie sich aus dem folgendem Abschnitt 4.2.3 ergibt, werden die meisten aller Datensätze für diese Arbeit nicht relevant sein, da sie keine Daten enthalten, die zur Ermittlung von Sehenswürdigkeiten in Köln gebraucht werden. Zusätzlich sollen die hier benötigten Datensätze auf ein Minimum beschränkt werden. Die Abfrage bleibt deshalb verständlicher. Ein Nachteil der auftreten könnte ist, dass die gefundenen Treffer nicht umfassend genug sind. Diesem soll entgegengewirkt werden, indem nach Möglichkeit Datensätze ausgewählt werden, die sehr viele RDF Tripel beinhalten.
Neben der Größe des Datensatzes spielt auch die Ontologie eine wichtige Rolle. Die Ontologie bestimmt die Spezialisierung des Datensatzes. Hier müssen Vokabeln definiert sein, aus denen sich potentielle Sehenswürdigkeiten ermitteln lassen. Welche Vokabeln wichtig sind, wird genauer in 4.3 behandelt. Die Ontologie wird für das SPARQL Query von zentraler Bedeutung sein.
Ein weiteres Kriterium für die Wahl einer Datensatzquelle ist ein existierender SPARQL Endpunkt.

4.2.2. Auswahlverfahren

Anhand der in Abschnitt 4.2.1 definierten Kriterien eines Datensatzes soll nun beschrieben werden, wie die Datensätze gefunden wurden. Auf den Internetseiten `https://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets` und `https://www.w3.org/wiki/DataSetRDFDumps` hat das W3C Listen von Datensätzen des Semantic Web zur Verfügung gestellt. Neben den Links zu den Datensätzen ist jeweils auch eine kurze Beschreibung über den Datensatz vorhanden. Anhand dieser Listen werden potentielle Kandidaten identifiziert.

4.2.3. Gefundene Datensatzquellen

In diesem Abschnitt werden die gefundenen Datensätze aufgelistet und kurz beschrieben. Mit diesen soll im weiteren Verlauf dieser Arbeit die Abfrage entwickelt werden:

1. **DBpedia** ist, wie anhand der Abbildung 3.4 zu erkennen, ein zentraler Datensatz im Semantic Web. DBpedia extrahiert strukturierte Informationen von Wikipe-

dia und stellt diese im RDF Format bereit. DBpedia ist zusätzlich in weiteren Sprachversionen als eigener Datensatz vertreten. In dieser Arbeit wird jedoch nur die englische Version betrachtet.
Status: Online - Zuletzt überprüft am 23.08.2016

2. **GeoNames** ist ähnlich wie DBpedia ein zentraler Datensatz im Semantic Web. GeoNames bietet geographische Informationen zu Ländern und Orten der Erde.
Status: Online - Zuletzt überprüft am 23.08.2016

3. **Wikidata** ist ein Datensatz der strukturierte Daten aus den Wikimedia-Projekten enthält.
Status: Online - Zuletzt überprüft am 23.08.2016

4. **YAGO** ist ein Datensatz, der ähnlich wie DBpedia Informationen direkt aus Wikipedia extrahiert, aber weniger RDF Tripel besitzt.
Status: Online - Zuletzt überprüft am 23.08.2016

Bei den gefundenen Datensätzen handelt es sich, mit Ausnahme von GeoNames, jeweils um Cross-Domain Datensätze. Das bedeutet, dass die Daten die verschiedensten Themengebiete abdecken. Spezialisierte Datensätze, die die Stadt Köln oder Sehenswürdigkeiten betreffen, konnten nicht identifiziert werden. Im Folgenden wird auch der GeoNames Datensatz nicht weiter berücksichtigt, um einheitlich mit Cross-Domain Datensätzen zu arbeiten.

4.2.4. Quantität der Daten

Alle hier aufgeführten Datensätze sind im einzelnen überdurchschnittlich groß.

- DBpedia hatte im Jahr 2014 etwa drei Milliarden Einträge. [Biz14].
- Wikidata gibt an, dass derzeit 15 Millionen Eintrage vorliegen, davon sollen 3,7% Bauwerke sein. [Wik16c].
- YAGO gibt an, dass der Datensatz über zehn Millionen Einträge besitzt [o.V14].

Summiert man die Einträge der Datensätze kommt man auf die beachtliche Zahl von 3,025 Milliarden Einträgen. Alleine die Wahrscheinlichkeit, dass aus diesen Einträgen die Sehenswürdigkeiten der Stadt Köln ermittelt werden, ist recht hoch. Hinzu kommt, dass die drei Datensätze DBpedia, Wikidata und YAGO ihre Daten unter anderem von Wikipedia beziehen. Eigene Nachforschungen haben ergeben, dass viele Sehenswürdigkeiten ihren eigenen Wikipediaartikel besitzen. Dazu wurde nach bekannten Bauwerken in Köln auf der englischen Wikipediaseite gesucht. Daher kann angenommen werden, dass sich diese auch in den Datensätzen wiederfinden.

4.2.5. Qualität und Validität der Daten

Wie in Abbildung 3.2 zu erkennen ist, spielt auch das Vertrauen eine wesentliche Rolle im Semantic Web. Jedem steht es offen Informationen als Linked Open Data öffentlich zur Verfügung zu stellen. Aus diesem Grund und dem der unüberschaubaren Menge an Daten, die alleine schon die hier drei vorgestellten Datensätze besitzen, ist es nahezu unmöglich die Daten auf ihre Korrektheit zu prüfen [GHO12, S. 21]. Die Autoren Guha et al. weisen darauf hin, dass Vielem im Internet nicht vertraut werden kann. Ein Computerprogramm ist nicht in der Lage alle im Internet als Fakten definierte Aussagen zu überprüfen [GMM03, S. 702]. Werden also Informationen aus dem Semantic Web benutzt existiert immer das Risiko, dass die Daten falsch oder inkonsistent zu anderen Datensätzen sind. Dem Anwender bleiben daher nur die Möglichkeiten, entweder darauf zu vertrauen, dass die Informationen nach bestem Wissen und Gewissen korrekt veröffentlicht wurden oder sie eigenständig zu überprüfen. Da die meisten Einträge der hier verwendeten Datensätze aus Wikipedia generiert werden, muss auch bei dieser Arbeit darauf vertraut werden, dass die Angaben korrekt sind. Wikipedia versucht durch eine eigene Initiative eine Grundqualität der Artikel zu garantieren. So unterliegen neue Artikel einer Eingangskontrolle. Für diese Arbeit kann die Qualität von Wikipedia aus den folgenden Punkten jedoch durchaus toleriert werden:

1. In dieser Arbeit wird lediglich nach Sehenswürdigkeiten gesucht. Kritische Attribute wie die Entstehungsgeschichte oder die heutige und damalige Bedeutung sind für diese Arbeit nicht von Belang.

2. Dank der Kontrolle, die Wikipedia anstrebt, kann darauf vertraut werden, dass Daten wie Entstehungsjahr, Ort oder Erbauer mit einer annehmbaren Fehlertoleranz korrekt sind. Es macht für diese Arbeit beispielsweise keinen Unterschied, ob der Bau des Kölner Doms im Jahr 1248, das korrekte Jahr oder im Jahr 1300 begann. Beide Jahreszahlen liegen innerhalb der Auswahlkriterien für die zu ermittelnden Sehenswürdigkeiten.

4.3. Vorbereitung der Abfrageentwicklung

In diesem Abschnitt wird der Vorbereitungsprozess der Entwicklung beschrieben. Die Abfrage stellt einen zentralen Teil dieser Arbeit dar. Zunächst soll in 4.3.1 die Vorgehensweise der Entwicklung genauer geschildert werden. Im nächsten Schritt werden die zur Umsetzung erforderlichen Vorbereitungen getroffen. Zu diesen Schritten zählen

unter anderem das Spezifizieren von SPARQL Endpunkten und das Herausfinden von benötigten Vokabeln und vorhandenen Datenressourcen der jeweiligen Datensätze.

4.3.1. Vorgehensweise

Da bei der Entwicklung des Abfrage keine Erfolgsgarantie gegeben ist, soll auch der Entwicklungsprozess möglichst offen gestaltet werden. Dadurch wird das Ergebnis variabler und die Wahrscheinlichkeit eines Erfolges oder Teilerfolges sollte erhöht werden.

Im Vorfeld müssen dennoch einige vorbereitende Schritte erarbeitet werden, um eine strukturierte und nicht willkürliche Entwicklung zu garantieren. Diese werden in den nächsten Unterabschnitten erklärt. Anschließend soll die Entwicklung beginnen.

Bei der Entwicklung sollen bereits vorhandene Lösungsmöglichkeiten wie Codebeispiele herangezogen werden, um schnell Resultate vorliegen zu haben. Durch die Kombination vieler solcher Codebeispiele und das eigenständige Anpassen an die eigenen Kriterien, wird die Abfrage mit der Zeit immer komplexer und präzisere Ergebnisse liefern. Es handelt sich um einen iterativen Entwicklungsprozess. Der Abschnitt 4.4 beschreibt die durchgeführten Iterationen.

Wie in den vorherigen Kapiteln bereits beschrieben, wird es nicht möglich sein genau zu erkennen, wann und ob das Ziel vollständig erreicht wurde. Im Semantic Web existieren zu viele Daten, um sie alle zu verarbeiten. Da es sich bei dem RDF um einen Graphen handelt, existieren auch hier Navigationsprobleme, die NP- oder Co-NP Vollständig sind [ZZLY02, vgl.] [Hor05, vgl.]. Des Weiteren sind die Interessen an einer Sehenswürdigkeit bei jedem Menschen unterschiedlich, sodass nur versucht werden kann, ein möglich genaues, aber kein perfektes Ergebnis zu erzielen. Daher soll der Entwicklungsprozess eingestellt werden, sobald sich beim Verfasser dieser Arbeit ein zufriedenstellendes Ergebnis eingestellt hat. Während der Entwicklung soll nicht ausgeschlossen werden, dass mehrere Abfragen entstehen.

In dem Abschnitt 4.6 werden Beobachtungen beschrieben, die während der Entwicklung aufgefallen sind. Mittels Literaturquellen sollen die hier erhaltenden Beobachtungen abgeglichen werden, um Gemeinsamkeiten oder Unterschiede festzustellen.

4.3.2. Geographische Koordinaten von Köln

Das Ermitteln von Sehenswürdigkeiten im Stadtgebiet Köln erfordert, dass dieser Raum näher spezifiziert wird. Eine genaue Lokalisierung eines geographischen Ortes kann

durch die Angabe der Längen- und Breitengrade durchgeführt werden. Hier muss berücksichtigt werden, dass das Stadtgebiet Köln zu groß ist, um es mit einem Koordinatenpaar als Punkt zu lokalisieren. Eine Fläche eignet sich besser.
Die Stadt Köln bietet auf ihrer Webseite einen Datensatz an, der das Stadtgebiet mittels geografischer Koordinaten exakt eingrenzt [Red16c]. Für weitere Vergleiche ist diese Eingrenzung jedoch zu umfangreich und soll daher vereinfacht werden. Es bietet sich ein Rechteck an, bestehend aus den maximalen und minimalen Koordinatenwerten. Die Tatsache, dass dadurch ein nicht unerheblicher Teil, der nicht zur Stadt Köln gehört, abgedeckt wird, soll für diese Arbeit vernachlässigt werden. Zum einen wären potentiell gefundene Sehenswürdigkeiten immer noch in unmittelbarer Nähe der Stadt Köln und zum anderen soll der Fokus dieser Arbeit nicht darauf liegen, das Standgebiet möglichst exakt zu untersuchen, sondern viel mehr, ob es möglich ist, Sehenswürdigkeiten in einem definierten Gebiet zu finden.
Die ermittelten minimal und maximal Koordinaten aus dem Datensatz lauten:

Längengrad: 6.7725304027693065° bis 7.1620279942184437°
Breitengrad: 50.830449396039278° bis 51.084974339607413°

Mittels dieser Koordinaten kann beispielsweise ermittelt werden, ob ein Objekt bei Köln liegt und somit als Sehenswürdigkeit in Frage kommt.

4.3.3. SPARQL Endpunkte

Über einen SPARQL Endpunkt kann der Anwender eine Abfrage an einen Datensatz abschicken. Je nach Anbieter des Endpunktes kann der Anwender die Darstellung der Resultate selbst auswählen. Zu den möglichen Formaten gehören unter anderem:

- Hypertext Markup Language (HTML)
- RDF/XML
- JSON
- N-Tripels

Auf der Webseite des YAGO Datensatzes unter `https://www.mpi-inf.mpg.de/de/departments/databases-and-information-systems/research/yago-naga/yago/demo/` wird ein SPARQL Endpunkt der Pariser Hochschule *Paris-Saclay* vorgeschlagen. Über diesen Dienst können alle hier vorgestellten Datensätze abgefragt werden.
Die Entwicklung der Abfrage wird hauptsächlich über diesen Endpunkt erfolgen. Verfügbar ist der Endpunkt unter der Adresse: `https://io.datascience-paris-saclay.`

`fr/exampleInsertUpdate.php`. Alle Abfragen der Entwicklungsiteration[1] sind über diesen Endpunkt ausführbar.

4.3.4. Vokabeln und Ressourcen

Grundsätzlich sollen im Vorfeld so wenig Vokabeln und Ressourcen wie möglich festgelegt werden. Dadurch existiert bei der Entwicklung mehr Spielraum für anderweitige Begriffe. Dennoch lohnt und empfiehlt es sich vorab bei den Datensatzanbietern zu informieren, mit welchen Vokabeln und Ressourcen potentielle Ergebnisse erzielt werden können.

DBpedia Unter der Adresse `http://mappings.dbpedia.org/server/ontology/classes/` ist die Ontologie von BDpedia einsehbar. Wichtiger Ausgangspunkt ist die Klasse `Place`. Von ihr werden unter anderem die Klassen `ArchitecturalStructure`, `PopulatedPlace` oder `NaturalPlace` abgeleitet. Besonders die Klasse `ArchitecturalStructure` und deren weitere Subklassen sind bei dieser Arbeit von wichtiger Bedeutung. Die folgenden Klassen haben das Potential Sehenswürdigkeiten, die die in 2.4 definierten Kriterien erfüllen, zu beschreiben:

- `ReligiousBuilding`
- `HistoricBuilding`
- `Castle`
- `Gate`
- `RouteOfTransportation`
- `Square`
- `Tower`

Neben den hier aufgeführten Klassen können noch weitere in Betracht kommen.

YAGO Unter der Adresse `https://www.mpi-inf.mpg.de/de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/` ist die Taxonomie von YAGO herunterladbar. Neben den verfügbaren Klassen, die sich aus Begriffen des Lexikons *WordNet* und einer ID-Nummer ergeben, sind in der Taxonomie auch Kategorien enthalten, die aus Wikipedia extrahiert wurden. Die Klassen sind vom Namen und ihrem Aufbau

[1] Die einzelnen Abfragen, ebenso wie eine kurze Gebrauchsanweisung, sind auf der Begleit-CD dieser Arbeit vorhanden.

vergleichbar mit BDpedia. Von den Kategorien konnten unter anderem die Folgenden identifiziert werden:

- `wikicat_Buildings_and_structures_in_Cologne`
- `wikicat_Churches_in_Cologne`
- `wikicat_Squares_in_Cologne`
- `wikicat_Places_of_worship_in_Cologne`
- `wikicat_Landmarks_in_Cologne`
- `wikicat_Monuments_and_memorials_in_Cologne`
- `wikicat_Cemeteries_in_Cologne`

Zwar besitzt auch DBpedia von Wikipedia extrahierte Kategorien, jedoch sind diese in der Ontologie nicht aufgeführt und müssen gezielt gesucht werden.
In dieser Arbeit sollen Kategorien jedoch kaum Verwendung finden und lediglich für Vergleiche benutzt werden. Das Verwenden dieser Kategorien kommt der in Abschnitt 4.1.2 beschriebenen navigierenden Suche sehr nahe. Die erforschende Suche erfordert einen höheren Abfrageaufwand und sollte daher aufschlussreichere Resultate liefern.

Wikidata Ein Dokument, welches die Ontologie von Wikidata aufzeigt, konnte nicht gefunden werden. Durch gezieltes Navigieren müssen die Ressourcen selbst ermittelt werden Ein Ansatz wäre das Attribut *owl:sameAs* bei den Datensätzen DBpedia und YAGO nach WikiData Einträgen abzusuchen.

4.3.5. SPARQL Output

An dieser Stelle soll vorab bereits eine Eingrenzung erfolgen, wie ein mögliches Abfrageergebnis aussehen könnte. Zwar können mit SPARQL viele verschiedene Dateninformationen ausgewählt werden, doch sollen bei dieser Abfrage nur allgemeine Informationen einer Sehenswürdigkeit projiziert werden. Für spätere Anwendungsprogramme läge hier ein wichtiger Konfigurationspunkt, um das Ergebnis den Interessen des Anwenders anzupassen.
Der Output der Abfrage soll eine tabellarische Struktur besitzen. Die Spalten werden aus den folgenden Einträgen bestehen.

Listing 4.1: Output Projektion der Abfrage

```
SELECT ?name ?location ?time ?image [...]
{
  [...]
} [...]
```

1. **Name** Die Angabe des Namens ist obligatorisch und soll die Sehenswürdigkeit eindeutig identifizieren.
2. **Ort** Unter dem Ort ist eine möglichst genaue Lokalisierung innerhalb der Stadt Köln zu verstehen. Es kann sich hierbei um Koordinaten oder eine Adresse handeln.
3. **Epoche/Zeit** Diese Spalte gibt an, wann das Bauwerk entstanden ist.
4. **Bild** Anstelle einer Bilddatei wird im Semantic Web viel mehr die URI des Bilds angegeben.

Die Spalten zwei bis vier sind optional, da nicht sichergestellt werden kann, dass die Informationen vorhanden sind.
Mit Hilfe der hier getroffenen Bestimmungen über die Darstellung des Resultats der Abfrage, lässt sich bereits ein erster Teil entwickeln. Listing 4.1 zeigt eine SPARQL Abfrage, die die in diesem Abschnitt beschriebene Tabelle als Resultat erzeugt.

4.3.6. Erwartete Resultate

Gemäß dem Prinzip der nachforschenden Suche, sollen bei der Entwicklung keine Sehenswürdigkeiten, mit Ausnahme des Kölner Doms, erwartet werden. Ein Resultat mit etwa 30 gefundenen Sehenswürdigkeiten wäre wünschenswert, soll dennoch nicht über den Erfolg oder Misserfolg entscheiden. Selbst wenn nach Betrachtung aller Kriterien keine Sehenswürdigkeiten gefunden werden sollten, lässt sich daraus die Erkenntnis gewinnen, dass der gewählte Ansatz nicht geeignet ist, um das Semantic Web für diese Fragestellung zu benutzen.

4.4. Entwicklungsiterationen

Der gesamte Entwicklungsprozess wurde in sieben Iterationen durchgeführt. Die siebte Iteration stellt den Abschluss der Entwicklung dar und wird genauer im Abschnitt 4.5 behandelt. Die Iterationen eins bis sechs werden in diesem Abschnitt kurz vorgestellt.

Dabei sollen der Inhalt der Iteration, der Abfragecode, sowie das Abfrageresultat beschrieben werden. Auf der Begleit-CD sind die einzelnen Iterationen mit Code und Ergebnis festgehalten. Der Code der siebten Iteration ist im Anhang B nachzulesen.

4.4.1. Erste und zweite Iteration

In den ersten zwei Entwicklungsiterationen soll sich mit der Abfragesprache SPARQL vertraut gemacht werden. Zu diesem Zweck wurden Abfragebeispiele, zu finden unter `https://www.w3.org/2009/Talks/0615-qbe/`, verwendet. Diese werden so abgeändert, dass sie dennoch in den Kontext dieser Arbeit passen.

In der ersten Iteration sollen deutsche Städte ermittelt werden, die mehr als eine Millionen Einwohner besitzen. Durchgeführt wird die Abfrage auf dem Datensatz von DBpedia. Ziel dieser Iteration war das Lernen des Umgangs mit Variablen und der `FILTER` Verwendung in einer SPARQL Abfrage. Zum einen ist die Abfrage dadurch länger, als nötig gewesen wäre und zum anderen ist sie wegen des eher unüblichen Vergleichs von Zeichenketten zur Bestimmung der Variable `?country_name` fehleranfälliger.[2] Beim Abfrageergebnis fällt auf, dass die Stadt Berlin nicht aufgeführt ist, weil Berlin im DBpedia Datensatz nicht vom Typ `City` ist [o.Va].

In der zweiten Iteration sollen feste Objekte wie Bauwerke in Deutschland ermittelt werden. Neben DBpedia wird auch der Datensatz von Wikidata abgefragt. Hier tritt bereits ein großer Unterschied der beiden Datensätze auf, der bis zur letzten Iteration bemerkbar bleibt. Bringt DBpedia insgesamt fast 14300 Ergebnisse, so liefert Wikidata mit über 52000 Einträgen mehr als das dreifache an Daten zurück.

4.4.2. Dritte und vierte Iteration

Mit der dritten und vierten Iteration sollte sich der Ermittlung der Kölner Sehenswürdigkeiten langsam genähert werden. Zunächst wird in der dritten Iteration die Anzahl der Bauwerke ermittelt, die sich in der Stadt Köln befinden. Als Bauwerke werden die Entitäten spezifiziert, die der Klasse `ArchitecturalStructure` unter DBpedia oder der Klasse `Construction` unter Wikidata angehören. Ob sich eine Entität im Stadtgebiet von Köln befindet, wird je nach Datensatz unterschiedlich gelöst.

> Bei **DBpedia** werden alle Bauwerke betrachtet, die die Attribute `isPartOf`, `location`, `locatedInArea` und `locale` besitzen und deren Werte auf die

[2] vgl. Zeile 25 der Datei *Code_ Iteration_ 1_ DBpedia.txt* auf der Begleit-CD

Entität der Stadt Köln verweisen. Zusätzlich werden mittels der in Abschnitt 4.3.2 ermittelten Koordinaten alle Bauwerke dem Ergebnis hinzugefügt, die sich in diesem Gebiet befinden. Dieser Teil der Abfrage ist in den Zeilen 18 bis 39 in Listing B.1 ablesbar. Es werden 147 Bauwerke ermittelt.

Bei **Wikidata** wird überprüft, ob das gefundene Bauwerk das Attribut `located in the administrative territorial entity` besitzt und ob dessen Wert der Entität `city part of Cologne` angehört. Die Zeilen 64 und 65 in Listing B.2 realisieren diese Ortsabfrage. Es werden 488 Bauwerke ermittelt.

Die gefundenen Lösungsmöglichkeiten werden in allen weiteren Iteration unverändert weiterverwendet.

In der vierten Iteration werden die vorhandenen Abfragen um den Zusatz erweitert, dass nur Bauwerke berücksichtigt werden sollten, die dem Kriterium „historisch und kulturell“ entsprechen. Entsprechende Klassen wurden bereits im Abschnitt 4.3.4 am Beispiel von DBpedia vorgestellt. Da DBpedia auch Klassen des YAGO Datensatzes bereitstellt, werden diese der DBpedia Abfrage hinzugefügt. Daraus ergeben sich die folgenden Klassen für eine DBpedia und YAGO Abfrage:

- **DBpedia:** `Castle`, `HistoricBuilding`, `ReligiousBuilding`, `Gate`, `Port`, `Bridge`, `Mill`, `Square`, `HistoricPlace` und `Monument`
- **YAGO:** `PlaceOfWorship`, `Memorial`, `PublicSquare`, `Gate`, `Bridge`, `Tower`, und `Port`

Auch wenn einige der Klassen einen identischen Namen besitzen, so ist es dennoch wichtig beide Klassen mit aufzunehmen. Dadurch können auch Entitäten gefunden werden, die lediglich einer der beiden möglichen Klassen zugeordnet sind. Die Zeilen 65 und 66 in Listing B.1 zeigen diese Abfrage. Die Abbildung 4.1 zeigt den Verlauf der gefundenen Sehenswürdigkeiten der DBpedia- und Wikidataabfrage ab der dritten Entwicklungsiteration. Die DBpedia und YAGO Abfrage ergibt insgesamt 19 Bauwerke, was einen Verlust von 128 Bauwerken im Vergleich zur dritten Iteration bedeutet.
Bei der Wikidata Abfrage werden vergleichbare Klassen verwendet:

`Church`, `Palace`, `Castle`, `Gate`, `Port`, `Mill`, `Square`, `Monument`, `Bridge` und `Tower`

Zusätzlich verfügt der Wikidata Datensatz über das Attribut `heritage status` welches den Wert `Baudenkmal` annehmen kann. In Voraussicht auf die weiteren Iterationen,

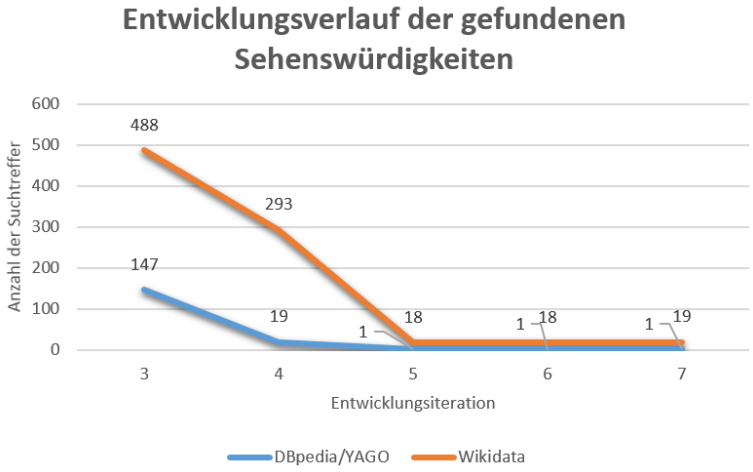

Abbildung 4.1.: Entwicklungsverlauf der gefundenen Sehenswürdigkeiten

werden bei dieser Abfrage alle in Köln liegenden Bauwerke aufgenommen, die dieses Attribut mit dem Wert `Baudenkmal` besitzen. Durch einen `UNION` Block, zu erkennen in den Zeilen 21 bis 25 aus Listing B.2, wird dieser Zusatz eingefügt. Aus der Wikidata Abfrage ergeben sich 293 Einträge, was einen Verlust von 195 Bauwerken im Vergleich zur dritten Iteration bedeutet. Prozentual gesehen fällt der Verlust bei Wikidata jedoch mit knapp 40% geringer aus als bei DBpedia mit ca. 87%.
Zusätzlich wird in dieser Iteration eine DBpedia und YAGO Abfrage entwickelt, die anstelle der Klassentypen Kategorien abfragt.[3] Dadurch können Vergleiche gezogen werden, die feststellen, in wie weit sich durch die Benutzung dieser Kategorien das Ergebnis verbessert oder verschlechtert. Die Kategorien, vorgestellt in 4.3.4, beziehen sich dabei direkt auf die Stadt Köln, wodurch dieser Teil der Abfrage ausgelassen werden konnte.[4] Diese Abfrage lieferte mit 61 Treffern weitaus mehr Ergebnisse.

4.4.3. Fünfte und sechste Iteration

In der fünften Iteration werden die Abfragen aus den Iterationen drei und vier mit dem Zusatz versehen, dass nur Bauwerke, die vor dem 19. Jahrhundert entstanden sind, berücksichtigt werden sollen. Dazu werden zunächst die Resultate der letzten Iteration genauer nach Attributen begutachtet, die eine entsprechende Zeitangabe angeben. Obwohl die beiden Datensätze verschiedene Attribute besitzen, die über diesen Sachverhalt Auskunft geben, ist das Vorgehen bei beiden Abfragen doch identisch:

[3] vgl. Zeilen 52 bis 84 der Datei *Code_Iteration_4_DBpedia.txt* auf der Begleit-CD
[4] vgl. die beiden Abfragen in der Datei *Code_Iteration_4_DBpedia.txt* auf der Begleit-CD

Zunächst werden alle bisher ermittelten Entitäten überprüft, ob sie eines der notwendigen Attribute besitzen. Sofern das Attribut existiert, wird dessen Wert in eine Variable gespeichert. Da die Variable für eine Entität überschrieben werden kann, ist es wichtig, dass Attribute, die das Entstehungsdatum beinhalten, erst nach Attributen, die beispielsweise nur das Datum der ersten Erwähnung beinhalten, abgefragt werden. Wird die Variable befüllt, werden durch einen `FILTER` Aufruf nur die Bauwerke ausgegeben, die vor dem 19. Jahrhundert datiert wurden.

Bei der DBpedia und YAGO Abfrage wird nach diesen Kriterien nur der Kölner Dom ermittelt.[5] Im Vergleich zur vierten Iteration ist ein Verlust von ungefähr 95% entstanden. Bei der Wikidata Abfrage reduzierte sich das Ergebnis um etwa 94% auf insgesamt 18 Treffer.

Die sechste Iteration berücksichtigt das Kriterium, dass keine Bauwerke aufgeführt werden sollen, die ein kommerzielles Veranstaltungsgebäude darstellen. Da die DBpedia und YAGO Abfrage zu diesem Zeitpunkt nur einen Treffer listet, der nicht in dieses Kriterium fällt, wurde die weitere Entwicklung hier ausgelassen. Bei der Wikidata Abfrage wird ein weiterer `FILTER` Aufruf hinzugefügt, der abfragt, ob die zu befragende Entität der Klassen `business enterprise`, `venue` oder `museum` und deren Unterklassen angehört. Die Zeilen 74 bis 80 in Listing B.2 zeigen den Code. Falls eine Entität vom Typ einer dieser Klassen ist, wird sie nicht aufgelistet. Das Ergebnis von 18 erzielten Bauwerken bleibt jedoch unverändert.

4.5. Entwickelte Abfrage

In diesem Abschnitt wird die siebte und letzte Iteration vorgestellt, in welcher die zwei ermittelten Abfragen erweitert werden. Zusätzlich werden die gefundenen Sehenswürdigkeiten etwas genauer betrachtet.

4.5.1. Siebte Iteration

Die Selektion der Daten wurde mit der sechsten Iteration abgeschlossen. In dieser Iteration sollte die Projektion der Daten die in 4.3.5 festgelegten Anforderungen erfüllen. Zu diesem Zweck wird das `SELECT` Statement mit neuen Variablen ergänzt, sodass der Name, der Ort, ein Entstehungsdatum und ein Bild dargestellt werden können. Innerhalb des `WHERE` Blocks wurden neue Variablen deklariert und mit Werten bestückt.

[5] vgl. Resultate in der Datei *Resultate_Iteration_5_DBpedia.txt* auf der Begleit-CD

Grundsätzlich sollen alle Angaben, bis auf den Namen der Sehenswürdigkeit, optional sein.
Die Ermittlung des Orts erfolgte in beiden Abfragen in Abhängigkeit des vorhandenen Codes. Bei der Wikidata Abfrage konnte die bereits deklarierte Variable `?location` verwendet werden. Durch den `SERVICE` Block, in der Zeile 68 in Listing B.2, wird diese Variable dann in `?locationLabel` automatisch als lesbarer Klartext umgewandelt. Eine andere Lösung wird bei der DBpedia und YAGO Abfrage gefunden. Hier muss die Variable `?location` zunächst erzeugt werden. In nachfolgenden `OPTIONAL` Blöcken wird die Variable beschrieben. Dieses Vorgehen ist notwendig, da nicht alle Entitäten einheitlich das gleiche Attribut verwenden. Ist kein Attribut vorhanden, welches das Bauwerk mit der Entität Köln verbindet, so wird stattdessen die Geo-Koordinate des Bauwerks verwendet und projiziert. Nachlesbar ist dieser Codeteil in den Zeilen 75 bis 78 in Listing B.1.
Durch einen `OPTIONAL` Block werden die Bildadressen einer Sehenswürdigkeit ermittelt, sofern sie vorhanden sind. Besitzt ein Bauwerk kein Bild, wird es dennoch aufgelistet. Mit Ausnahme der unterschiedlichen Attributnamen der verschiedenen Datensätze, ist der Code in beiden Abfragen gleich, wie die Zeilen 81 in Listing B.1 und 82 in Listing B.2 zeigen. Die Werte für die Entstehungszeit wurden bereits in den vorherigen Iterationen ermittelt und können ohne weitere Änderung für die Projektion übernommen werden. Da einige Sehenswürdigkeiten mehrere Bilder oder Daten zum Entstehungsdatum besitzen können, werden zunächst für die gleiche Sehenswürdigkeit auch so viele Einträge erzeugt, wie Bilder und Zeitangaben vorhanden sind. Um diese Duplikate zu vermeiden, wird die Aggregatfunktion `SAMPLE` der `SELECT` Klausel hinzugefügt. Die `SAMPLE` Funktion wählt dabei zufällig einen Wert der Wertemenge aus. Die Verwendung einer Aggregatfunktion erfordert letztlich auch eine `GROUP BY` Anweisung für alle nicht aggregierten Variablen am Ende des `WHERE` Blocks [HS13].

4.5.2. Ermittelte Sehenswürdigkeiten

4.5.2.1. DBpedia und YAGO Abfrage

Schon nach der fünften Iteration erzielte die DBpedia und YAGO Abfrage nur eine Sehenswürdigkeit - den Kölner Dom. Tabelle 4.1 zeigt den Output der Abfrage. Bedenkt man, dass DBpedia als der größte öffentliche Datensatz in der Linked Open Data Cloud bekannt ist, so könnte das erzielte Ergebnis als ernüchternd betrachtet werden. Die beispielhafte Verwendung von fest definierten Kategorien aus der vierten Iteration erzielte zwar mehr Treffer, doch kann angenommen werden, dass das Ergebnis nach dem Hinzufügen weiterer Ausschlusskriterien rapide sinkt. Gründe für diese starke Abnahme sind vor allem:

1. Quantität der Dateneinträge. Obwohl DBpedia der größte Datensatz der Linked Open Data Cloud ist, sind deren Einträge der englischen Version von Wikipedia entnommen. Viele deutsche Bauwerke besitzen keinen englischsprachigen Wikipediaeintrag und sind somit nicht in DBpedia vertreten. Die Verwendung der deutschen Version von DBpedia besitzt wahrscheinlich mehr Einträge und würde zumindest bis zur dritten Iteration dementsprechend mehr Einträge liefern.

2. Vernachlässigen von Attributen. Sowohl DBpedia, als auch YAGO besitzen eine Vielzahl von Attributen mit denen Ressourcen beschrieben werden können. Jedoch werden sehr viele Attribute nicht einer Ressource zugeordnet. Notwendige, sowie optionale Suchkriterien können daher nicht abgefragt werden. Eine Ressource ist somit nicht auffindbar und wird durch das Setzen dieser Kriterien nicht als Ergebnis aufgenommen.

3. Inkonsistenz bei der Verwendung von Attributen. Beide Datensätze bieten nicht nur sehr viele Attribute an, sondern einige Attribute haben ähnliche Bedeutungen. Bei einer Abfrage müssen all diese Attribute berücksichtigt und auf Existenz in der Ressource geprüft werden. Der Code der Abfrage wird dadurch um viele Zeilen und Operationen wie beispielsweise durch die `UNION` Verknüpfung länger.

Tabelle 4.1.: Sehenswürdigkeiten unter Auslass der Bildadresse (DBpedia & YAGO)

Name der Sehenswürdigkeit	**Ort**	**Entstehungszeit**
"Kölner Dom"@de	"50.9413 6.958"	"1248"

4.5.2.2. Wikidata Abfrage

Bei der Wikidata Abfrage werden 19 Sehenswürdigkeiten ermittelt, die der Tabelle 4.2 zu entnehmen sind. Der Gewinn einer Sehenswürdigkeit im Vergleich zur sechsten Iteration ist auf ein Duplikat zurückzuführen, welches sich auf Grund der Verwendung des `SERVICE` Blocks, der die lesbaren Namen einer Variablen bestimmt, nicht eliminieren lässt. Die gefundenen Sehenswürdigkeiten zeichnen sich dennoch durch die folgenden Merkmale aus:

1. Der Kölner Dom ist nicht unter den gefundenen Sehenswürdigkeiten. Grund dafür ist die fehlende Existenz einer Angabe zur Entstehungszeit [Wik16b]. Auch wenn der Wikidata Datensatz mehr Ergebnisse liefert als die DBpedia und YAGO Abfrage, so können die oben aufgeführten Begründungspunkte aus 4.5.2.1 zwei und drei auch auf Wikidata überführt werden. Punkt eins trifft nicht zu, da der

Wikidata Datensatz aus allen verfügbaren Sprachversionen von Wikipedia erstellt wird.

2. Obwohl viele Bauwerkarten als Ergebnis möglich wären, sind neun von neunzehn Treffer katholische Kirchen. Einige Bauwerkarten wie alte Mühlen oder öffentliche Plätze werden hingegen nicht aufgelistet. Der hohe Kirchenanteil ist auf die christlich geprägte Geschichte von Köln zurückzuführen.

3. Alle aufgeführten Sehenswürdigkeiten stammen aus dem Mittelalter, obwohl auch Bauwerke der Antike oder frühen Neuzeit zulässig gewesen wären. Damit verbunden ist auch, dass viele Sehenswürdigkeiten in der Altstadt von Köln liegen. Dennoch sind auch Gebäude außerhalb des Stadtkerns vertreten wie zum Beispiel die Burg Worringen.

4. Sofern im Datensatz nur das Entstehungsjahr angegeben ist, wird dieser Wert als der erste Januar des Jahres interpretiert.

5. Obwohl bereits in der sechsten Iteration kommerzielle Veranstaltungsgebäude herausgefiltert werden sollten, finden sich im Endresultat das Gaffel Haus und der Gürzenich wieder. Grund ist ein fehlender Eintrag, die diese Bauwerke zusätzlich als Veranstaltungsgebäude kennzeichnen [Wik15a] [Wik15b].

6. Die Burg Worringen existiert heutzutage gar nicht mehr [Tü13] und kann daher eigentlich nicht als Sehenswürdigkeit aufgeführt werden. Es existiert jedoch kein Attribut auf Wikidata, mit dem sich das hätte herausfiltern lassen [Wik16a].

4.6. Entwicklungsbeobachtungen

In den folgenden Abschnitten werden Beobachtungen festgehalten, die im Verlauf der Entwicklung der SPARQL Abfrage gemacht wurden. Während der Literaturarbeit wurden bereits Feststellungen und Erfahrungen anderer Autoren erarbeitet, die in den folgenden Untersektionen vorgestellt und mit den eigenen verglichen werden.

4.6.1. Datenduplikate

Bei der Entwicklung der Abfrage werden ausschließlich Cross-Domain Datensätze verwendet. Sowohl Wikidata, als auch DBpedia beziehen und generieren ihre Daten von der Internetenzyklopädie Wikipedia. Es liegt daher nahe, dass die Datensätze jeweils eigene Ontologien und Ressourcen besitzen, die dennoch das gleiche real existierende Objekt

Tabelle 4.2.: Sehenswürdigkeiten unter Auslass der Bildadresse (Wikidata)

Name der Sehenswürdigkeit	**Ort**	**Entstehungszeit**
Burg Worringen	Köln-Worringen	Jan 1, 1275
St. Aposteln	Köln-Altstadt-Nord	Jan 1, 1020
Hahnentorburg	Köln-Altstadt-Nord	Jan 1, 1300
Hahnentorburg	Köln-Altstadt-Süd	Jan 1, 1300
Alt St. Pankratius	Köln-Worringen	Jan 1, 1150
Rathaus Köln	Köln-Altstadt-Nord	Jan 1, 1200
Alte Dorfkirche	Junkersdorf	Jan 1, 1223
St. Amandus	Köln-Merkenich	Jan 1, 1200
Bayenturm	Köln-Altstadt-Süd	Jan 1, 1300
Frankenturm (Köln)	Köln-Altstadt-Nord	Jan 1, 1200
St. Maria im Kapitol	Köln-Altstadt-Süd	Jan 1, 1100
Gürzenich	Köln-Altstadt-Nord	Jan 1, 1500
Gaffel Haus	Köln-Altstadt-Nord	Jan 1, 1213
Severinstorburg	Köln-Altstadt-Süd	Jan 1, 1300
St. Andreas	Köln-Altstadt-Nord	Jan 1, 0974
Nikolaus-Kapelle	Westhoven	Jan 1, 1100
Antoniterkirche	Köln-Altstadt-Nord	Jan 1, 1350
Ulrepforte	Köln-Altstadt-Süd	Jan 1, 1300
Alt St. Alban	Köln-Altstadt-Nord	Jan 1, 1150

beschreiben. Das Semantic Web bietet dank der OWL und dem Attribut `owl:sameAs` die Möglichkeit die beiden Datensatzressourcen miteinander zu verbinden. Bei DBpedia findet dieses Attribut eine häufige Verwendung. Als Beispiel wird hier die Ressource des Kölner Doms, zu erreichen unter `http://dbpedia.org/page/Cologne_Cathedral`, betrachtet. Das `owl:sameAs` Attribut verweist neben den anderen Sprachversionen von DBpedia auch auf die Wikidata Ressource des Doms. Allerdings lässt sich daraus nicht schließen, dass auch Wikidata zurück auf DBpedia verweist. Ein Blick auf die entsprechende Wikidata Ressource zeigt, dass DBpedia nicht angegeben wird. Genauer betrachtet wird nicht einmal das Attribut `owl:sameAs` verwendet. Als Berners-Lee die Grundregeln von Linked Data vorstellte, betonte er gleichermaßen direkt, dass das Einhalten dieser Regeln nicht verpflichtend sei [BL09]. So ist auch die Verwendung von `owl:sameAs` mehr optional als obligatorisch.

Auch wenn mittels `owl:sameAs` auf gleichartige Ressourcen verwiesen werden kann, so ist die Annahme, dass es sich um reine Duplikate handelt nicht differenziert genug. So untersuchen die Autoren Halpin et al. in ihrem Artikel die Verwendung dieses Attributes. Sie kommen zu dem Ergebnis, dass das `owl:sameAs` Attribut nicht nur bei identischen, sondern auch bei ähnlichen Ressourcen verwendet wird. Sie ermitteln, dass in maximal 72% aller Fälle das Attribut korrekt verwendet wird [HHM+10, S. 317].

Vergleicht man die gespeicherten Informationen von DBpedia und Wikidata, so kann festgestellt werden, dass durchaus Unterschiede in Bezug auf vorhandene Attribute und Attributwerte entstehen [HHM+10, S. 30]. Solche Inkonsistenzen treten im Semantic Web häufig auf, wie diese Arbeit zeigt. Weiteres wird in Abschnitt 4.6.2 beschrieben.

4.6.2. Konsistenz der Daten

Eine existierende Datenkonsistenz spielt in der Datenverarbeitung eine erfolgsentscheidende Rolle. Daten, die willkürlich und ohne einheitliche Form existieren, lassen sich nur mit großem Aufwand verarbeiten. Das Resource Definition Framework stellt eine Grundstruktur der Daten im Semnatic Web dar. Ohne das RDF wären Anwendungen mit Linked Data nicht möglich. Für das Abfragen der Daten mittels SPARQL sind jedoch noch weitere Kriterien entscheidend.
Obwohl DBpedia und Wikidata ihre Informationen aus Wikipedia beziehen, sind deren Ontologien sehr ähnlich, jedoch nicht identisch. Weitere Inkonsistenzen finden sich bei der Verwendung von Attributen. Während DBpedia Attribute aus anderen Ontologien wie OWL oder YAGO zulässt, werden äquivalente, jedoch eigene Attribute bei Wikidata verwendet. DBpedia verlinkt Vokabeln mittels des `owl:equivalentProperty` Attributes nach Wikidata [IKA+15]. Umgekehrt erfolgt dieser Schritt nicht. Diese Heterogenität macht das Abfragen von Daten aufwändig [JHY+10, S. 83].

Listing 4.2: Abfrage zur Ermittlung deutscher Bauwerke unter Angabe ihrer Entstehungszeit

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX type: <http://dbpedia.org/ontology/>
3  PREFIX prop: <http://dbpedia.org/property/>
4
5  SELECT distinct ?place_name ?time
6  WHERE
7  {
8    ?place rdf:type/rdfs:subClassOf* type:Place;
9    rdfs:label ?place_name ;
10   type:country <http://dbpedia.org/resource/Germany>.
11
12   ?place type:buildingStartDate ?time.
13
14   FILTER (lang(?place_name) = "de")
15 } Limit 7
```

Weitere Inkonsistenzen finden sich bei den verschiedenen Sprachversionen des DBpedia Datensatzes. So existieren hier nicht einheitliche Attributbezeichnungen, sondern jede Sprachversion besitzt ihre eigenen. Dieser heterogene Datensatzaufbau lässt nur bedingt eine Interoperabilität der verschiedenen Sprachausführungen zu.
Jedoch treten nicht nur zwischen den Datensätzen Inkonsistenzen auf, sondern auch innerhalb diesen. Einige in dieser Arbeit verwendeten Attribute können verschiedenartige Werte annehmen. Das DBpedia Attribut `buildingStartDate` kann beispielsweise mehrere Darstellungsformen besitzen. In dem Codebeispiel 4.2 wird in Zeile 12 die Entstehungszeit eines deutschen Bauwerks ermittelt. Die Ausgabe ist in Tabelle 4.3 dokumentiert. Deutlich zu erkennen sind die verschiedenen Zeitformate, die ein Bauwerk besitzt. Neben dem reinen Jahr sind auch Zeitspannen und andere Formen vertreten. Grund für diese inkonsistente Benutzung ist der definierte Wertetyp des Attributs, der durch `rdfs:range` als `xsd:string` festgelegt wird. Da in einer Zeichenkette beliebige Informationen gespeichert werden können, ist die Zahl der möglichen Zeitformate praktisch unbegrenzt. Eine Werteabfrage kann daher sehr aufwendig werden. Es empfiehlt sich daher, die zu verwendenden Attribute bereits während der Entwicklung auf diese Eigenschaft hin zu überprüfen.

Tabelle 4.3.: Resultat der Abfrage aus dem Codebeispiel 4.2

place_name	**time**
"Burg Falkenstein (Pfronten)"@de	"1270-1280; 1885"
"Eschbachtalsperre"@de	"1889"
"Schloss Lichtenstein (Württemberg)"@de	"1840"
"Hasselvorsperre"@de	"1956"
"Mandelholztalsperre"@de	"1952"
"Oderteich"@de	"1715"
"Talsperre Wippra"@de	"February 1951"

Wie in Abschnitt 4.6.1 schon angedeutet, sind nicht alle Attribute gleichermaßen bei den Ressourcen eines bestimmten Typs vorhanden. Dadurch lässt sich die große Abnahme der Trefferergebnisse in der fünften Iteration im Vergleich zur Vierten erklären, welche in 4.1 zu erkennen ist. Die entsprechenden Attribute, die eine Entstehungszeit angeben würden, sind bei den meisten aller Ressourcen nicht vorhanden. Wird die zeitliche Filterung, die Listing 4.3 zeigt, in der fünften Iteration auskommentiert, erzielt die Abfrage 57 Treffer.

Listing 4.3: Zeitfilter, der nur Treffer zulässt, die vor dem 19. Jahrhundert lagen

```
1 FILTER (?date <= "1800-01-01T00:00:00Z"^^xsd:dateTime)
```

Inkonsistenzen sind für eine zu entwickelnde Abfrage von wichtiger Bedeutung, da sie einerseits Möglichkeiten bieten, dass mehr und verschiedene Informationen gesammelt werden können, andererseits entsteht jedoch ein größerer Entwicklungsaufwand um diese Unterschiede zu berücksichtigen. In dieser Arbeit werden ab der dritten Iteration durch das Hinzufügen mehrerer `UNION` Operationen die Entitäten nach verschiedenen Attributen durchsucht, um den Ort eines Bauwerks zu identifizieren. Zu nennen wäre hier beispielsweise die Implementation der Abfrage zur Entstehungszeit eines Bauwerks in den Zeilen 41 bis 61 in Listing B.1.

4.6.3. Manipulationsaufwand

Aus den bisher geschilderten Erkenntnissen lassen sich auch Aussagen zum Manipulationsaufwand der Daten treffen. Unter dem Manipulationsaufwand soll in dieser Arbeit der Aufwand verstanden werden, der aufgebracht wurde, um aus den verwendeten Datensätzen und Technologien eine Abfrage mit Resultat zu erstellen. Zusammengefasst ist die Arbeit mit dem Semantic Web zum Einstieg mit einen hohen Zeit- und Rechercheaufwand verbunden. In welchen Punkten sich diese Aufwände widerspiegeln wird nachfolgend aufgezeigt.

Obwohl das Semantic Web das Ziel hat, seine Inhalte maschinenlesbar bereitzustellen, ist der Mensch dennoch wesentlicher Bestandteil der Datenverarbeitung. Die Autoren Jain et al. bemerken, dass es beispielsweise keinen Mechanismus gibt, welcher den Datensatz GeoNames als geografischen Datensatz beschreibt [JHY+10, S. 82]. Es liegt also in der Verantwortung eines Menschen, den richtigen Datensatz für die Problemstellung auszuwählen. Nach Ansicht von Jain et al. stellt das einen großen Nachteil dar. Anwendungen des Semantic Web sollten selbständig in der Lage sein diese Informationen zu beschaffen [JHY+10, S. 82]. Daraus resultiert auch, dass sich der Entwickler mit den verschiedenen Datensätzen vertraut machen muss. Für ein einfaches Anwendungsszenario müssen zwei bis drei Datensätze in Betracht gezogen werden [JHY+10, S. 83]. In dieser Arbeit werden die Cross-Domain Datensätze von BDpedia mit Kombination der YAGO Klassen und Wikidata verwendet.

Des Weiteren kann die Auswahl der Datensätze durch das oftmals falsch benutze Attribut `owl:sameAs`, wie es in 4.6.1 beschrieben wurde, erschwert werden. Findet der Entwickler auf verschiedenen Datensätzen zwei durch `owl:sameAs` verknüpfte Ressourcen, welche unterschiedliche Informationen bezüglich eines Attributes wie dem Entstehungsdatum besitzen, kann ein Entscheidungskonflikt entstehen [JHY+10, S. 83].

Wie im Abschnitt 4.6.2 beschrieben, existieren von DBpedia mehrere Sprachausführungen. Jede Version besitzt eigene Attribute, die in der jeweiligen Landessprache benannt sind. Sehenswürdigkeiten aus Deutschland sind in der deutschen Version von DBpedia ausführlicher mit Daten bestückt, als beispielsweise auf dem englischen oder französischen Datensatz. Will ein Entwickler diesen Vorteil nutzen, ist er gezwungen, für jede Sprachversion, die er verwenden will, eine eigene Abfrage zu erstellen. Hinzu kommt, dass er unter Umständen auch die Landessprache des Datensatzes verstehen muss, um Attribute einfacher zu identifizieren. Dieser Schritt wäre auch mittel des `owl:sameAs` Attributs möglich. In beiden Fällen wird dem Entwickler jedoch ein größer Entwicklungsaufwand abverlangt.

Neben dem Wissen um die Existenz der Datensätze, müssen für eine Abfrage auch speziellere Kenntnisse des Datensatzes erworben werden. Die Ontologie, Klassen, Attribute, deren mögliche Werte, sowie die in 4.6.2 beschriebenen Inkonsistenzen eines Datensatzes müssen dem Entwickler bekannt sein [JHY+10, S. 83]. Verwendet DBpedia noch sprechende Attributnamen wie `buildingStartDate`, so nutzt Wikidata numerische Identifikationsnummern. Die Präfixe **Q** und **P** geben an, ob es sich um ein Attribut (P) oder um eine Entität (Q) handelt. Diese Identifikationsnummern erschweren das Erlernen des Datensatzes und das Lesen des Abfragecodes, sofern keine Kommentare vorhanden sind, die diese Identifikationsnummern in lesbare Namen übersetzen.
Diese Unterschiede machen es ebenfalls erforderlich, dass für jeden Datensatz eine eigene Abfrage entwickelt wird.

Neben der Auseinandersetzung mit den Datensätzen, erfordert auch die Entwicklung der SPARQL Abfrage einiges an Aufwand. SPARQL bietet eine Vielzahl an verfügbaren Operationen, die einzeln gesehen einfach erscheinen. Die Kombinationsmöglichkeit dieser Operationen machen SPARQL jedoch zu einer komplexen Abfragesprache, die es zunächst zu verstehen gilt [PAG06, S. 30 f.]. Hinzu kommt, dass die Syntax der Sprache es erfordert, dass der Entwickler seine Suchkriterien genau spezifizieren muss. Auch hier wird vorhandenes Wissen des betreffenden Datensatzes vorausgesetzt [JHY+10, S. 83].

4.6.4. Wieder- und Weiterverwendbarkeit

Das Semantic Web wird kontinuierlich durch neue Informationen, Daten und Verknüpfungen angereichert. Daher ist es möglich, und sehr wahrscheinlich, dass ein existierendes SPARQL Query mit der Zeit unterschiedliche Resultate ausgibt. Es können dabei sowohl positive, als auch negative Effekte auftreten. Neue Ressourcen können dem Suchergebnis hinzugefügt werden oder alte werden nicht mehr aufgeführt. Gründe können

das Hinzufügen oder Löschen alter Ressourcen sein. Des Weiteren können neue Attribute einer Ressource hinzugefügt werden oder der Wert eines Attributs wird geändert. Je nach Inhalt der Abfrage erscheinen neue oder es verschwinden alte Treffer.
Im Verlauf dieser Arbeit konnte dieses Phänomenen nicht beobachtet werden. Es ist jedoch möglich, dass die beiden entwickelten Abfragen in Zukunft mehr Ergebnisse liefern. Wie die Resultate der Iterationen vier und fünf zeigen, findet mit dem Hinzufügen des Kriteriums der Entstehungszeit ein erheblicher Verlust an Treffern statt. Abbildung 4.1 zeigt diesen Verlust. Grund ist hauptsächlich, wie in 4.6.2 beschrieben, dass die gefundenen Ressourcen aus der vierten Iteration keine derartige Angabe besitzen. Sollten diese in Zukunft der Ressource hinzugefügt werden, kann auch mit mehr Treffern gerechnet werden.
Sofern die Attribute, Klassen oder Ressourcen einer Abfrage nicht aus dem Datensatz vollständig entfernt werden, kann die Abfrage jederzeit wiederverwendet werden.

Die Syntax von SPARQL erlaubt das Kombinieren von algebraischen Operationen und `SELECT` Blöcken. Eine Abfrage kann dadurch sehr komplexe Strukturen annehmen. Durch `UNION` Blöcke und `FILTER` Anweisungen kann eine Abfrage jederzeit um weitere Suchkriterien und Anzeigekriterien ergänzt werden. `UNION` oder `MINUS` Blöcke können Abfragen auch in Abschnitte unterteilen, ohne das ein alter Code zu sehr verändert werden muss. In dieser Arbeit wurden `UNION` Blöcke hauptsächlich dazu verwendet, eine Menge von Bauwerken zu ermitteln, die mindestens eines von vielem möglichen Attributen besitzen, welche zur Entstehungszeit oder dem Ort Auskunft geben.
Auch die Verwendung von Variablen erweist sich als großer Vorteil von SPARQL. So ist es möglich bei der Abfrage nach der Bauwerksart die Variable `?type` zu deklarieren. Durch einen zusätzlichen `FILTER` Operator wird der Variable eine Menge von Klassen zugewiesen. Der Typ eines Bauwerks wird mit den Werten der Variable abgeglichen. Gibt es eine Übereinstimmung, entspricht das Bauwerk dem Suchkriterium. In der 17. Zeile in Listing B.2, wird diese Technik unter anderem angewendet. Diese Menge von Klassen lässt sich im Code einfach neuen Suchkriterien anpassen.
Für Anwendungsprogramme, die diesen Abfragecode verwenden, stellen die Variablen eine geeignete Schnittstelle zur Nutzerinteraktion dar, da sich die Bedürfnisse eines Nutzer besser durch Variablen ändern lassen.

4.6.5. Automatisierungsmöglichkeiten

In diesem Abschnitt wird beschrieben, welche Automatisierungsmöglichkeiten SPARQL liefert. Diese Eigenschaft ist besonders für sich ändernde Anwendungssituationen wichtig. Solche Entwurfsmuster, auch generische Abfragen genannt, haben den Vorteil,

dass sie viele Anwendungsszenarien unterstützen [CAJP93, S. 195]. SPARQL bietet unter gewissen Voraussetzungen die Möglichkeit, Abfragen automatisiert zu generieren. Welche Voraussetzungen existieren und wie eine solche Generierung aussehen würde, soll hier kurz beschrieben werden.

Ein wichtiger bereits angesprochener Punkt ist, dass der Entwickler einer Abfrage den Datensatz einerseits sehr gut kennen muss. Andererseits gibt es keine Möglichkeit mit der sich die Datensätze selbst beschrieben. Um eine Menge von Anwendungsszenarien generisch lösen zu können müssen dem Entwickler sehr viele detaillierte Informationen vorliegen. Das Entwickeln einer generischen Abfrage wird dadurch erschwert, denn potentielle Spezialfälle und Inkonsistenzen müssen berücksichtigt werden. Abstraktionen, wie sie bei solchen Entwurfsmustern erwartet werden [HC07, S. 1], sind daher schwer umsetzbar.
Die Syntax von SPARQL erlaubt jedoch einen gewissen Grad der automatischen Generierung. Die Verwendung von Variablen, wie sie in 4.6.4 erklärt wurde, lässt sich weiter ausführen. So könnte eine Abfrage beispielsweise durch das Anlegen konstanter Klassenmengen auf einen speziellen Anwendungsfall angepasst werden. Eine Softwarekomponente würde aus den von Entwicklern bereitgestellten Informationen eine Abfrage erstellen können. Die Softwarekomponente würde aus den vorhandenen Klassenmengen und anderen möglichen Variablen, die für diese Aufgabe passenden Werte entnehmen und in den Abfragecode einsetzten. Anders als bei modernen Programmiersprachen wie C# oder Java, wo generische Typen erst zur Laufzeit bestimmt werden, würde in diesem Fall die Abfrage zunächst erstellt und erst danach ausgeführt werden.

4.6.6. Skalierbarkeit

Die Frage der Skalierbarkeit ist bei Abfragesprachen wie SQL oder SPARQL von wichtiger Bedeutung. In dieser Arbeit werden sich die Fragen gestellt, ob für eine komplexe Abfrage genügend Daten vorhanden sind und ob eine komplexe Abfrage auf dem Datensatz überhaupt durchgeführt werden kann.

In den bisherigen Abschnitten 4.6.1 und 4.6.2 wurden Inkonsistenzen der Daten im Semantic Web beschrieben und welche Auswirkungen sie auf das Suchergebnis haben. Besonders die Tatsache, dass viele Ressourcen einige Attribute klassenübergreifend nicht einheitlich besitzen, lässt die Frage aufkommen, ob komplexe Suchanfragen überhaupt ausreichend beantwortet werden können. Auch die Autoren Jain et al. stellen sich diese Frage und führen als Grund unter anderem auch solche Inkonsistenzen auf. Ihrer Ansicht nach besteht die Linked Open Data Cloud größtenteils aus grundlegenden RDF

Tripeln. Das Potential, welches die OWL und das RDFS bieten, wird ihrer Meinung nach nicht ausgeschöpft. Sollten hier keine Verbesserungen auftreten, so befürchten sie, wird das Semantic Web lediglich ein Raum mit einfach nur vielen Daten bleiben [JHY+10, S. 82 f.].
Während der Entwicklung der Wikidata Abfrage in der dritten Iteration, sollte durch einen zusätzlichen `UNION` Block eine Umkreissuche durchgeführt werden. Das Listing 4.4 zeigt den `UNION` Block mit einem Suchradius von 30 Kilometern. Zusätzlich wurden zehn Kilometer gewählt. Daneben erfolgte die Abfrage mit dem `UNION` Code in der fünften Iteration. Das Ergebnis zeigt die Tabelle 4.4. Ein „X“ als Anzahl bedeutet Time Out Fehler. Alle Suchanfragen sind mit einem Time Out Fehler gescheitert. Den genauen Grund des Fehlers zu ermitteln, ist nicht Teil dieser Arbeit, jedoch sollen einige Möglichkeiten kurz aufgeführt werden:

Tabelle 4.4.: Resultat der erweiterten Umkreissuche im Wikidata Datensatz.

Iteration	**Anzahl der Treffer mit erweitertem UNION Block**	**Suchradius in km**
3	X	10
5	X	10
3	X	30
5	X	30

Listing 4.4: Wikidata Umkreissuche von Bauwerken in Köln

```
[...]
}
UNION
{
  wd:Q365 wdt:P625 ?cologneLoc .
  SERVICE wikibase:around
  {
    ?item wdt:P625 ?location .
    bd:serviceParam wikibase:center ?cologneLoc .
    bd:serviceParam wikibase:radius "30" .
  }
  # ?item ist ein Bauwerk
  ?item wdt:P31/wdt:P279* wd:Q811430.
  }
}
```

- Es gibt zu viele Treffer, die in der Abfragezeit nicht alle ermittelt werden konnten.

- Durch den `UNION` Operator wird die Abfrage komplexer und verbraucht mehr Ressourcen.
- Der Code ist nicht gut genug optimiert.
- Die Verwendung des `SERVICE` Operators verbraucht zu viele Ressourcen.

Der hier vorgestellte `UNION` Block fand aufgrund des Time Out Fehlers keine weitere Verwendung in den Entwicklungsiterationen. Eine Möglichkeit der Entlastung der SPARQL Abfrage wäre, dass `FILTER` oder `OPTIONAL` Operationen, sowie die Darstellung der Resultate in eine Anwendungssoftware verschoben werden.
Michael Hausenblas kritisiert in einem Artikel ebenfalls die Skalierbarkeit. Er bemängelt, dass es derzeit keine Anwendungen im Semantic Web gibt, die seriöse Probleme lösen. Die meisten Anwendungen beruhen auf ausgedachten, vereinfachten Problemszenarien und kleinen künstlichen Datensätzen. Anhand dieser könne die Skalierbarkeit nicht demonstriert werden [Hau09, S. 68]. Er stellt jedoch fest, dass die Skalierbarkeit bei dem Navigationsprinzip „ follow-your-nose “ noch nicht abschließend geklärt ist [Hau09, S. 71]. Bei diesem Navigationsprinzip werden Ressourcen im Semantic Web inspiziert, indem Verlinkungen schrittweise gefolgt werden. Da dieses Prinzip auch mit SPARQL durchführbar ist, ist die Frage der Skalierbarkeit auch hier noch offen. Auch Fernandez et al. sehen einen kritischen Punkt. Sie merken an, dass bei einer Suche im Semantic Web primär Instanzen einer Ontologie, anstatt von Dokumente ausgegeben werden. Durch die Abwesenheit einer Bewertungsmethode, wie sie beispielsweise Google besitzt, können die Suchergebnisse bei vielen Treffen nicht gut verarbeitet werden [FLS+08, S. 253]. Besonders passende Suchergebnisse können also nicht von weniger passenden Ergebnissen unterschieden werden. Für eine SPARQL Abfrage trifft diese Aussage jedoch nicht vollständig zu, da durch diverse Filter- und Aggregatoperationen die Ergebnisse angepasst werden können.

5. Ausblick und Fazit

In dieser Arbeit wurde die Entwicklung einer SPARQL Abfrage zur Ermittlung von Sehenswürdigkeiten in Köln in der Linked Open Data Cloud behandelt. Bevor in diesem Kapitel die wichtigsten Erkenntnisse des Entwicklungsprozesses noch einmal zusammengefasst werden, soll ein kurzer Ausblick auf die zukünftige Weiterarbeit mit dem Thema beschrieben werden.

Mit der Entwicklung einer, oder wie in diesem Falle zweier Abfragen, sind jedoch noch nicht alle offenen Fragen beantwortet und Forschungsmöglichkeiten ergründet. In zukünftigen Arbeiten könnten die hier erzielten Ergebnisse für die folgenden Zwecke weiterverwendet werden:

1. Der Vergleich mit weiteren Datensätzen. In dieser Arbeit wurde beispielsweise der Datensatz GeoNames nicht verwendet, um gezielter Resultate aus Cross-Domain Datensätzen miteinander zu vergleichen. Auch andere Datensätze, die eher weniger dem Thema „Sehenswürdigkeiten“ zuzuordnen sind, könnten in weiteren Untersuchungen genauer betrachtet werden. Zum einen könnte festgestellt werden, ob Datensätze aus anderen Domänen mehr oder weniger Treffer erzielen und zum anderen könnte festgestellt werden, ob die in dieser Arbeit ermittelten Schwächen des Semantic Web weitreichend aufzufinden sind.

2. Die Untersuchung der hier ermittelten Abfragen für andere Städte in Deutschland, oder allgemeiner für einen beliebigen Ort auf der Erde. In einer solchen Fortführungen könnte untersucht werden, ob andere Orte mehr Sehenswürdigkeiten hervorbringen als Köln. Besonders englische, oder amerikanische Großstädte wie London, oder New York versprechen großes Potential.

3. Durch die Betrachtung mehrerer Nutzungsszenarien könnte zudem untersucht werden wie groß der Modifikationsaufwand der Abfragen wäre, um diese Szenarien zufriedenstellend abzudecken. Der Punkt der Skalierbarkeit einer Abfrage könnte ebenfalls weiter beleuchtet werden.

4. Die Entwicklung eines Entwurfsmusters zur allgemeinen Bestimmung touristischer Punkte. In dem Abschnitt 4.6.5 wurde bereit das Prinzip einer generischen

Abfrage erwähnt. In weiteren Entwicklungsprojekten könnte untersucht werden in wie weit sich die hier entwickelten Abfragen abstrahieren lassen, um somit ein allgemein brauchbares Muster zu entwickeln.

5. Die Entwicklung einer Anwendungssoftware. Diese Arbeit konzentriert sich auf die Entwicklung einer Abfrage. Die Abfrage selbst ermittelt jedoch nur die Ergebnisse. Eine anwendungsbezogene Verarbeitung dieser Ergebnisse muss durch eine weitere Software gelöst werden. Auch die hier genannten Punkte eins bis vier könnten in einer solchen Software untergebracht werden. Einige Autoren bemängeln, dass es gerade für das Semantic Web zu wenige solcher Anwendungen gibt [GHO12, S. 21] [Hau09, S. 68]. In dem Artikel „Linked Open Data: The Essentials“ beschrieben die Autoren eine Vorgehensweise wie Linked Open Data Anwendungen entwickelt werden können [BK11, S. 36 f.].

Im zweiten Kapitel wurden zunächst Grundlagen über den Begriff der Sehenswürdigkeit vorgestellt. Es wurde festgestellt, dass für jeden Menschen andere Bauwerke, oder Punkte in einer Stadt von Interesse sind. Die möglichen Anwendungsszenarien für eine derartige Abfrage sind somit vielfältig.

Zur Ermittlung der Sehenswürdigkeiten konnten in der Linked Open Data Cloud keine speziellen Datensätze zu den Domänen „Sehenswürdigkeiten“ oder „Bauwerke“ gefunden werden. Auf Grund der Beobachtung, dass viele Sehenswürdigkeiten auf Wikipedia vertreten sind, wurde sich für Cross-Domain Datensätze entschieden, die ihre Daten aus Wikipedia generieren.

Obwohl mit der Stadt Köln als Anwendungsbeispiel eine Stadt gewählt wurde, die dem Autor dieser Arbeit vertraut ist, sollten Sehenswürdigkeiten alleine durch die Angabe der Suchkriterien ermittelt werden. Wären im Vorfeld schon Sehenswürdigkeiten festgelegt worden, hätte das nachforschende Suchprinzip im Semantic Web nicht untersucht werden können. Die entwickelten Abfragen dieser Arbeit zeigen deutlich, dass die nachforschende Suche im Semantic Web bei diesem Anwendungsszenario Ergebnisse liefern. Daraus kann geschlossen werden, dass das Semantic Web für alltägliche Aufgaben wie das Finden von Bauwerken an einem Ort, durchaus geeignet sein kann.

Eine ausführlichere Betrachtung der Ergebnisse aus den Entwicklungsiterationen zeigt jedoch einige Mängel auf.

1. Mit jeder weiteren Iteration wurden immer weniger Suchtreffer gefunden, wie Abbildung 4.1 zeigt. Dieses Verhalten war natürlich durch das Hinzufügen von

weiteren Kriterien zu erwarten. Abnahmen von über 90% der Suchergebnisse, nur durch das Hinzufügen einer Abfrage, ob eine Entstehungszeit angegeben ist, sind dennoch sehr groß und werfen Fragen nach dem Grund dieses Verlusts auf.

2. Das Endergebnis der siebten Entwicklungsiteration führt insgesamt 20 Treffer auf, von denen einer ein nicht eliminierbares Duplikat ist. Je nach Anwendungsszenario scheint diese Anzahl groß genug. Für einen Touristen, der beispielsweise nur einen Tag in der Stadt ist, könnten diese 20 Bauwerke reichen. Jemand der länger in der Stadt bleibt und sich sehr intensiv mit dem alten Köln auseinandersetzen möchte, könnte von dem Ergebnis enttäuscht sein.

3. Große und bekannte Bauwerke und Plätze der Stadt, die in die Suchkriterien fallen, sind nicht unter den Suchtreffern. Als Beispiele sind hier die Plätze *Neumarkt*, *Alter Markt*, oder *Heumarkt* zu nennen. Diese haben laut Wikipediaartikel ihren Ursprung schon zum Teil zur Römerzeit und erfüllen somit die Suchkriterien [Wik16f] [Wik16d] [Wik16e]. Ebenfalls ist die Kirche *Groß St. Martin* nicht unter den Suchergebnissen.

Ein identifizierter Hauptgrund für diese Mängel sind Inkonsistenzen, die zwischen den Datensätzen und innerhalb der Daten vorliegen. Wichtige Attribute, die die Suchkriterien beschreiben würden, sind bei vielen Ressourcen nicht vorhanden, oder besitzen uneinheitliche Darstellungsformen der Werte. Eine einheitliche Überprüfung fällt somit schwer und Sonderfälle müssen genau betrachtet werden. Der Entwickler einer Abfrage muss diese Besonderheiten kennen. Zudem muss er im Auge behalten, ob der Abfragecode überhaupt ausgeführt werden kann und nicht in Time Out Fehlern endet. Der Entwicklungsaufwand steigt daher an.

In dieser Arbeit wurden anhand eines Anwendungsszenario die Möglichkeiten und Schwächen des Semantic Web aufgezeigt. Grundsätzlich ist dieser Teil des Internets in der Lage, Aufgaben und Probleme des Alltags zu lösen. Viele kleine Mängel führen jedoch dazu, dass die Vision, die Berners-Lee beschreibt [BHBL09], nicht in dem erwarteten Maße zur Wirklichkeit wird. Es existieren bereits viele Ansätze und Ideen zur Behebung dieser Mängel. Da es laut Berners-Lee jedoch keine festen Regeln, sondern nur Empfehlungen gibt [BL09], kann es für deren Umsetzung und allgemeine Anerkennung jedoch keine Garantie geben.

Literaturverzeichnis

[BBDR+13] Bechhofer, Sean ; Buchan, Iain ; De Roure, David ; Missier, Paolo ; Ainsworth, John ; Bhagat, Jiten ; Couch, Philip ; Cruickshank, Don ; Delderfield, Mark ; Dunlop, Ian u. a.: Why linked data is not enough for scientists. In: *Future Generation Computer Systems* 29 (2013), Nr. 2, S. 599–611

[BBLPC14] Beckett, David ; Berners-Lee, Tim ; Prud'hommeaux, Eric ; Carothers, Gavin: RDF 1.1 Turtle / W3C. 2014. – W3C Recommendation. – https://www.w3.org/TR/2014/REC-turtle-20140225/

[Bec11] Beckett, Dave: *Re: What does SPARQL stand for?* `https://lists.w3.org/Archives/Public/semantic-web/2011Oct/0041.html`. Version: 2011, Abruf: 16.08.2016

[Bec14] Beckett, David: RDF 1.1 N-Triples / W3C. 2014. – W3C Recommendation. – https://www.w3.org/TR/2014/REC-n-triples-20140225/

[BHBL09] Bizer, Christian ; Heath, Tom ; Berners-Lee, Tim: Linked data-the story so far. In: *Semantic Services, Interoperability and Web Applications: Emerging Concepts* (2009), S. 205–227

[BHS03] Becker, Christoph ; Hofinger, Hans ; Steinecke, Albrecht: *Geographie der Freizeit und des Tourismus: Bilanz und Ausblick*. Oldenbourg Wissenschaftsverlag, 2003. – ISBN 3486274643

[Biz14] Bizer, Christian: *DBpedia Version 2014 released*. `https://web.archive.org/web/20141120120008/http://blog.dbpedia.org/2014/09/09/dbpedia-version-2014-released/`. Version: 2014, Abruf: 23.08.2016

[BK11] Bauer, Florian ; Kaltenböck, Martin: Linked open data: The essentials. In: *Edition mono/monochrom, Vienna* (2011)

[BL00] Berners-Lee, Tim: *Weaving the Web: The Past, Present and Future of the World Wide Web by its Inventor*. 2000

[BL09] Berners-Lee, Tim: *Linked Data*. `https://www.w3.org/DesignIssues/LinkedData.html`. Version: 2009, Abruf: 06.09.2016

[BLHL+01] Berners-Lee, Tim ; Hendler, James ; Lassila, Ora u. a.: The semantic web. In: *Scientific american* 284 (2001), Nr. 5, S. 28–37

[BLK+09] Bizer, Christian ; Lehmann, Jens ; Kobilarov, Georgi ; Auer, Sören ; Becker, Christian ; Cyganiak, Richard ; Hellmann, Sebastian: DBpedia-A crystallization point for the Web of Data. In: *Web Semantics: science, services and agents on the world wide web* 7 (2009), Nr. 3, S. 154–165

[Bra06] Bratt, Steve: *Semantic Web and Other W3C Technologies to Watch.* https://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/W3CTechSemWeb.pdf. Version: 2006, Abruf: 15.08.2016

[CAJP93] Cimino, James J. ; Aguirre, Anthony ; Johnson, Stephen B. ; Peng, Ping: Generic queries for meeting clinical information needs. In: *Bulletin of the Medical Library Association* 81 (1993), Nr. 2, S. 195

[CJ14] Cyganiak, Richard ; Jentzsch, Anja: *The Linking Open Data cloud diagram.* http://lod-cloud.net/versions/2014-08-30/lod-cloud.png. Version: 2014, Abruf: 18.08.2016

[DJK04] Dosta, Wolfgang ; Jeckle, Mario ; Kriechbaum, Wernerl: Ontologie (n). In: *Java Spektrum* (2004), Nr. 3, 51–54. http://www.jeckle.de/semanticWebServices/vokont.html, Abruf: 16.08.2016

[Dud] Dudenredaktion ; Dudenverlag, Bibliographisches I. (Hrsg.): *Sehenswürdigkeit.* http://www.duden.de/rechtschreibung/Sehenswuerdigkeit, Abruf: 10.08.2016

[Efr12] Efrati, Amir: Google Gives Search a Refresh. In: *The Wall Street Journal* (2012). http://www.wsj.com/articles/SB10001424052702304459804577281842851136290, Abruf: 22.08.2016

[Fei09] Feigenbaum, Lee: *SPARQL By Example.* https://www.w3.org/2009/Talks/0615-qbe/l. Version: 2009, Abruf: 16.08.2016

[FLS+08] Fernandez, Miriam ; Lopez, Vanessa ; Sabou, Marta ; Uren, Victoria ; Vallet, David ; Motta, Enrico ; Castells, Pablo: Semantic search meets the web. In: *Semantic Computing, 2008 IEEE International Conference on* IEEE, 2008, S. 253–260

[Fp] Farlex ; partners ; Wörterbuch., TheFreeDictionary.com D. (Hrsg.): *Sehenswürdigkeiten.* http://de.thefreedictionary.com/Sehenswürdigkeiten, Abruf: 10.08.2016

[GHO12] Gradmann, Stefan ; Hennicke, Steffen ; Olensky, Marlies: Linked Data. In: *cms-journal* 35 (2012)

[GMM03] Guha, Ramanathan ; McCool, Rob ; Miller, Eric: Semantic search. In: *Proceedings of the 12th international conference on World Wide Web* ACM, 2003, S. 700–709

[Hau09] Hausenblas, Michael: Exploiting linked data to build web applications. In: *IEEE Internet Computing* 13 (2009), Nr. 4, S. 68–73

[HC07] Henninger, Scott ; Corrêa, Victor: Software pattern communities: Current practices and challenges. In: *Proceedings of the 14th Conference on Pattern Languages of Programs* ACM, 2007, S. 14

[Hes02] Hesse, Wolfgang: Ontologie (n). In: *Informatik-Spektrum* 25 (2002), Nr. 6, S. 477–480

[HHM+10] Halpin, Harry ; Hayes, Patrick J. ; McCusker, James P. ; McGuinness, Deborah L. ; Thompson, Henry S.: When owl: sameas isn't the same: An analysis of identity in linked data. In: *International Semantic Web Conference* Springer, 2010, S. 305–320

[Hor05] Horst, Herman J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3 (2005), Nr. 2, S. 79–115

[HS13] Harris, Steven ; Seaborne, Andy: SPARQL 1.1 Query Language / W3C. 2013. – W3C Recommendation. – http://www.w3.org/TR/2013/REC-sparql11-query-20130321/

[IKA+15] Ismayilov, Ali ; Kontokostas, Dimitris ; Auer, Sören ; Lehmann, Jens ; Hellmann, Sebastian: Wikidata through the Eyes of DBpedia. In: *arXiv preprint arXiv:1507.04180* (2015)

[JHY+10] Jain, Prateek ; Hitzler, Pascal ; Yeh, Peter Z. ; Verma, Kunal ; Sheth, Amit P.: Linked Data Is Merely More Data. In: *AAAI Spring Symposium: linked data meets artificial intelligence* Bd. 11, 2010

[Las99] Lassila, Ora: Resource Description Framework (RDF) Model and Syntax Specification / W3C. 1999. – W3C Recommendation. – http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[Lat02] Latz, Wolfgang: *Diercke Erdkunde für Gymnasien in Nordrhein-Westfalen : 5. Schuljahr, Neuausgabe*. first. Braunschweig : Westermann Schulbuchverlag GmbH, 2002. – ISBN 3141142750

[Lex07] Lexikon, TERRA ; Klett (Hrsg.): *Städtetourismus*. `http://www2.klett.de/sixcms/list.php?page=lexikon_suchergebnis_artikel&extra=Terra-online&inhalt=&mytitle=Geographie%20Lexikon&titelfamilie=&artikel_id=183425`. Version: 2007, Abruf: 10.08.2016

[LS05] Landgrebe, Silke ; Schnell, Peter: *Städtetourismus*. Oldenbourg Verlag, 2005

[Mar06] Markoff, John: Entrepreneurs See a Web Guided by Common Sense. In: *The New York Times* (2006). `http://www.nytimes.com/2006/11/12/business/12web.html`, Abruf: 15.08.2016

[Mil98] Miller, Eric: An introduction to the resource description framework. In: *Bulletin of the American Society for Information Science and Technology* 25 (1998), Nr. 1, S. 15–19

[O'r07] O'reilly, Tim: What is Web 2.0: Design patterns and business models for the next generation of software. In: *Communications & strategies* (2007), Nr. 1

[o.Va] o.V. ; DBpedia (Hrsg.): *About: Berlin.* `http://dbpedia.org/page/Berlin`, Abruf: 01.09.2016

[o.Vb] o.V. ; W3Schools.com (Hrsg.): *XML RDF.* `http://www.w3schools.com/xml/xml_rdf.asp`, Abruf: 17.08.2016

[o.V09] o.V. ; Consortium, World Wide W. (Hrsg.): *Platform for Internet Content Selection (PICS).* `https://www.w3.org/PICS/`. Version: 2009, Abruf: 17.08.2016

[o.V14] o.V. ; Informatik, Max Planck I. (Hrsg.): *YAGO: Overview.* `http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/`. Version: 2014, Abruf: 23.08.2016

[o.V15] o.V. ; W3C (Hrsg.): *What is a Vocabulary?* `https://www.w3.org/standards/semanticweb/ontology`. Version: 2015, Abruf: 30.08.2016

[o.VoJ] o.V. ; Kernforschung (CERN), Europäische O. (Hrsg.): *The birth of the World Wide Web.* `http://timeline.web.cern.ch/timelines/the-birth-of-the-world-wide-web`. Version: o.J., Abruf: 15.09.2016

[PAG06] Pérez, Jorge ; Arenas, Marcelo ; Gutierrez, Claudio: Semantics and Complexity of SPARQL. In: *International semantic web conference* Springer, 2006, S. 30–43

[Red16a] Redaktion ; Köln, Stadt (Hrsg.): *Köln - Sehenswürdigkeiten.* `http://www.arcgis.com/home/webmap/viewer.html?useExisting=1&layers=59221f7b8a5f4fefafdf5c0f597419ef`. Version: 2016, Abruf: 11.08.2016

[Red16b] Redaktion ; Köln, Stadt (Hrsg.): *Sehenswürdigkeiten in Köln.* `http://www.offenedaten-koeln.de/dataset/sehenswürdigkeiten-köln`. Version: 2016, Abruf: 11.08.2016

[Red16c] Redaktion ; Köln, Stadt (Hrsg.): *Stadtgebiet Koeln.* `http://www.offenedaten-koeln.de/dataset/stadtgebiet-koeln`. Version: 2016, Abruf: 01.09.2016

[Sch10] Schürig, Henning: *Social Media statt Web 2.0.* `http://www.henningschuerig.de/2010/social-media-statt-web-20/`. Version: 2010, Abruf: 15.08.2016

[Spi04] Spivack, Nova: *New Version of My "Metaweb" Graph — The Future of the Net.* http://www.novaspivack.com/science/new-version-of-my-metaweb-graph-the-future-of-the-net. Version: 2004, Abruf: 15.08.2016

[Tü13] Tückmantel, Ulli: Entrepreneurs See a Web Guided by Common Sense. In: *RP Online* (2013). http://www.rp-online.de/nrw/der-verlust-und-der-jammer-waren-gross-aid-1.3436535l, Abruf: 06.09.2016

[Wik15a] Wikidata: *Q1491042 — Wikidata,.* https://www.wikidata.org/w/index.php?title=Q1491042&oldid=266988705. Version: 2015, Abruf: 06.09.2016

[Wik15b] Wikidata: *Q153450 — Wikidata,.* https://www.wikidata.org/w/index.php?title=Q153450&oldid=278298755. Version: 2015, Abruf: 06.09.2016

[Wik16a] Wikidata: *Q1014663 — Wikidata,.* https://www.wikidata.org/w/index.php?title=Q1014663&oldid=291493626. Version: 2016, Abruf: 06.09.2016

[Wik16b] Wikidata: *Q4176 — Wikidata,.* https://www.wikidata.org/w/index.php?title=Q4176&oldid=361856828. Version: 2016, Abruf: 06.09.2016

[Wik16c] Wikidata: *Wikidata:Statistics.* https://www.wikidata.org/wiki/Wikidata:Statistics. Version: 2016, Abruf: 23.08.2016

[Wik16d] Wikipedia: *Alter Markt (Köln) — Wikipedia, Die freie Enzyklopädie.* https://de.wikipedia.org/w/index.php?title=Alter_Markt_(K%C3%B6ln)&oldid=157740632. Version: 2016, Abruf: 09.09.2016

[Wik16e] Wikipedia: *Heumarkt (Köln) — Wikipedia, Die freie Enzyklopädie.* https://de.wikipedia.org/w/index.php?title=Heumarkt_(K%C3%B6ln)&oldid=156321846. Version: 2016, Abruf: 09.09.2016

[Wik16f] Wikipedia: *Neumarkt (Köln) — Wikipedia, Die freie Enzyklopädie.* https://de.wikipedia.org/w/index.php?title=Neumarkt_(K%C3%B6ln)&oldid=157235363. Version: 2016, Abruf: 09.09.2016

[ZZLY02] Zhu, Haiping ; Zhong, Jiwei ; Li, Jianming ; Yu, Yong: An Approach for Semantic Search by Matching RDF Graphs. In: *FLAIRS Conference*, 2002, S. 450–454

Anhang

A. Umfrage: Sehenswürdigkeiten in Köln

In dieser kleinen Umfrage wurden neun Personen befragt, was ihrer Ansicht nach die zweitwichtigste Sehenswürdigkeit - nach dem Kölner Dom - von Köln ist. Mit dieser Umfrage sollte überprüft werden, in wie weit das persönliche Interesse dazu beitragt, dass Sehenswürdigkeiten anders gewichtet werden. Die Umfrage wurde im persönlichen Umfeld mündlich durchgeführt. Im Anschluss werden die Antworten der Personen aufgelistet. In den eckigen Klammern steht die Häufigkeit des Vorkommens der Antwort:

- Alter Markt [1]
- Altstadt [3]
- Groß St. Martin [1]
- Hohenzollernbrücke [1]
- Kranhäuser [1]
- Rhein-Energie Stadion [1]
- Rheinufer [1]

B. Listings der siebten Entwicklungsiteration

In diesem Anhang finden sich die Abfragecodes der siebten Entwicklungsiteration. Mit dieser Iteration wurde der Entwicklungsprozess in dieser Arbeit beendet. Die hier aufgeführten Codes sind daher das Endresultat und sollen sowohl in der Arbeit als auch auf der Begleit-CD verfügbar sein. Aufgrund der Länge der Codes wurden diese in den Anhang verschoben.

Da der Entwicklungsprozess iterativ war und die siebte Iteration auf den Iterationen drei bis sechs basiert, können auch diese älteren Iterationen auf die Codes verweisen.

Listing B.1: Abfragecode der siebten Iteration (DBpedia und YAGO)

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3  PREFIX type: <http://dbpedia.org/ontology/>
4  PREFIX prop: <http://dbpedia.org/property/>
5  PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
6  PREFIX georss: <http://www.georss.org/georss/>
7  PREFIX yago: <http://dbpedia.org/class/yago/>
8  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
9
10 SELECT distinct ?name ?location (SAMPLE(?time) AS ?time) (SAMPLE(?image) AS ?image)
11 WHERE
12 {
13   {
14     ?place rdf:type/rdfs:subClassOf ?type;
15     rdfs:label ?name.
16   }
17   # Verwende Entitaeten, die in Koeln liegen
18   {
19     ?place type:location ?city.
20     ?city type:isPartOf <http://dbpedia.org/resource/Cologne>.
21   }
22   union
23   {
24     ?place type:location <http://dbpedia.org/resource/Cologne>
25   }
26   union
27   {
28     ?place type:locatedInArea <http://dbpedia.org/resource/Cologne>
```

```
29  }
30  union
31  {
32    ?place prop:locale <http://dbpedia.org/resource/Cologne>
33  }
34  union
35  {
36    ?place geo:lat ?lat.
37    ?place geo:long ?long.
38    FILTER (?lat > 50.830449396039278 && ?lat < 51.084974339607413 && ?long > 6.7725304027693065
        ↪  && ?long < 7.1620279942184437)
39  }
40  #ermittle die Entstehungsdaten
41  {
42    {
43      ?place prop:yearcompleted ?time
44    }
45    union
46    {
47      ?place prop:founded ?time
48    }
49    union
50    {
51      ?place type:buildingStartDate ?time
52    }
53    union
54    {
55      ?place prop:dateOpened ?time
56    }
57    union
58    {
59      ?place type:openingDate ?time
60    }
61  }
62
63  # Suche nach Entitaeten der BDpedia-Klassen 'Castle', 'HistoricBuilding', 'ReligiousBuilding',
        ↪  'Gate', 'Port', 'Bridge', 'Mill', 'Square', 'HistoricPlace' und 'Monument'
64  # Suche nach Entitaeten der YAGO-Klassen 'PlaceOfWorship', 'Memorial', 'PublicSquare', 'Gate',
        ↪  'Bridge', 'Tower' und 'Port'
65  FILTER(?type IN (type:Castle, type:HistoricBuilding, type:ReligiousBuilding, type:Gate,type:
        ↪ Port, type:Bridge,type:Mill,type:Square, type:HistoricPlace,type:Monument,
66  yago:PlaceOfWorship103953416, yago:Memorial103743902, yago:PublicSquare108619620, yago:
        ↪ Gate103427296, yago:Bridge102898711, yago:Tower104460130, yago:Port108633957))
67
68  # Liste nur die deutschen Namen auf
69  FILTER (lang(?name) = "de")
70
71  # Nur Entitaeten vor dem 19. Jahrhundert
72  FILTER (xsd:integer(?time) <= 1800)
73
74  # Gibt den Ort an. Zunaechst nur die Koordinaten. Sind fuer einen Menschen besser lesbare Orte
        ↪  angegeben wird der alte Wert ueberschrieben
75  OPTIONAL { ?place georss:point ?location}
76  OPTIONAL { ?place type:locale ?location}
```

```
  OPTIONAL { ?place type:locatedInArea ?location}
  OPTIONAL { ?place type:location ?location}

  # Gibt das Bild an.
  OPTIONAL { ?place type:thumbnail ?image}
} group by ?name ?location
```

Listing B.2: Abfragecode der siebten Iteration (Wikidata)

```
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX ps: <http://www.wikidata.org/prop/statement/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT distinct ?itemLabel ?locationLabel (SAMPLE(?time) AS ?time) (SAMPLE(?image) AS ?image)
WHERE
{
  {
    {
      # Suche nach Entitaeten der Klassen 'church', 'palace', 'castle', 'gate', 'port', 'mill', '
          ↪ square', 'monument', 'bridge' und 'tower'
      ?item wdt:P31/wdt:P279* ?type.
      FILTER(?type IN (wd:Q16970, wd:Q16560, wd:Q23413, wd:Q53060, wd:Q44782, wd:Q44494, wd:
          ↪ Q174782, wd:Q4989906, wd:Q12280, wd:Q12518))
    }

    #Fuege zusaetzliche alle Entitaeten der KLasse Construction hinzu, die ein Baudenkmal sind
    UNION
    {
      ?item wdt:P31/wdt:P279* wd:Q811430.
      ?item wdt:P1435 wd:Q15632117.
    }
  }
  #Ermittle die Entstehungsdaten
  {
    {
      ?item wdt:P1619 ?time
    }
    union
    {
      ?item p:P793 ?eventstatement.

      ?eventstatement ps:P793 ?event.
      ?eventstatement pq:P580 ?time.
    }
    union
    {
      ?item p:P31 ?eventstatement.

      ?eventstatement ps:P31 ?event.
```

```
    ?eventstatement pq:P580 ?time.
  }
  union
  {
    ?item p:P31 ?eventstatement.

    ?eventstatement ps:P31 ?event.
    ?eventstatement pq:P1249 ?time.
  }
  union
  {
    ?item wdt:P580 ?time
  }
  union
  {
    ?item wdt:P571 ?time
  }
}

# Verwende Entitaeten, die in Koeln liegen
?item wdt:P131 ?location.
?location wdt:P31/wdt:P279* wd:Q15632166;

# Liste nur die deutschen Namen auf
SERVICE wikibase:label { bd:serviceParam wikibase:language "de" }

# Nur Entitaeten vor dem 19. Jahrhundert
FILTER (?time <= "1800-01-01T00:00:00Z"^^xsd:dateTime)

# Keine Entitaeten, der Klassen 'business enterprise', 'venue' und 'museum'
FILTER (NOT EXISTS{

  ?item wdt:P31/wdt:P279* ?excludeType.

  FILTER(?excludeType IN (wd:Q4830453, wd:Q17350442, wd:Q33506))

})

OPTIONAL {?item wdt:P18 ?image}
} group by ?itemLabel ?locationLabel
```

C. Inhalt der Begleit-CD

Auf der Begleit-CD sind alle Abfragecodes der einzelnen Iterationen und deren Resultate, alle in dieser Arbeit verwendetet Bilder, sowie die Arbeit selbst im PDF Format enthalten. Zur Verwendung der Abfragecodes wurde ein erklärendes Textdokument auf der Begleit-CD im .txt Format angelegt.

Der Verzeichnisbaum der Begleit-CD ist in Listing C.1 angegeben:

Listing C.1: Verzeichnisbaum der Begleit-CD

```
Begleit-CD
   +---Arbeit
   |   Bachlorarbeit_Sebastian_Leuer.pdf
   +---Grafiken
   \---Quellcode
       |  Readme.txt
       |  All in one Code und Resultate.txt
       +---1. Iteration
       +---2. Iteration
       +---3. Iteration
       +---4. Iteration
       +---5. Iteration
       +---6. Iteration
       \---7. Iteration
```

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 16. September 2016

Sebastian Leuer